Reward Optimizing Recommendation with Deep Learning and Maximum Inner Product Search

Criteo AI Lab KDD Tutorial 2022

Imad Aouali, Amine Benhalloum, Martin Bompaire, Achraf Ait Sidi Hammou, Sergey Ivanov, Benjamin Heymann, David Rohde, Otmane Sakhi, Flavian Vasile, Maxime Vono

Introduction – The Practical Stuff

What are we trying to do ?

Recommend the best items to a user

- Recommend: Within the context of online advertising, low latency constraints, slates
- Best: Maximizing a given reward (clicks, sales, …)
- Items: Tens of millions of products
- User: History of past user interactions, adapt to change in behaviour quickly, users enter and leave the system continually

Score millions of items very efficiently while adapting to changes in user behaviour

General framework

We are interested in decision rules of the form: $\boldsymbol{score}(\boldsymbol{u}, \boldsymbol{a}) \sim \boldsymbol{u}^{\boldsymbol{T}} \boldsymbol{\beta}_{\boldsymbol{a}}$

$\boldsymbol{u}$ – ”user embedding” of dimension D

$\boldsymbol{\beta}$ – The item embeddings of dimension $D \times P$

β_a – The embedding of item a

General framework

We are interested in decision rules of the form: $score(u, a) \sim \boldsymbol{u^T \beta_a}$

Why ?

To find the best action we need:

$$a^* = \text{argmax}x_{a' \in \{1,..,P\}}\, score(u, a')$$

If P is large, then this is very slow for general $score$ functions.
Using Dot Product enables us to leverage (Approximate) Maximum Inner Product Search techniques.

General framework

We are interested in decision rules of the form: $score(u, a) \sim \boldsymbol{u^T \beta_a}$

Why ?

What happens during training ? We can't evaluate each action, negative sampling techniques come in handy here.

General framework

We are interested in decision rules of the form: $score(x, a) \sim f_{\Xi}(\boldsymbol{x})^T \boldsymbol{\beta_a}$

$\boldsymbol{x}$ – The “context” the information we use to personalize on.

f_{Ξ} - Mapping from “context” to embedding

General framework

We are interested in decision rules of the form: $score(x, a) \sim f_{\Xi}(\boldsymbol{x})^T \boldsymbol{\beta_a}$

Why ?

- Users enter and leave the system constantly
- We want to adapt to change in behaviour
- Generalize across users

General framework

We are interested in decision rules of the form: $score(x, a) \sim f_{\Xi}(\boldsymbol{x})^T h_{\beta}(\boldsymbol{a})$

$\boldsymbol{a}$ – Set of initial pre-trained embeddings, product attributes and features …

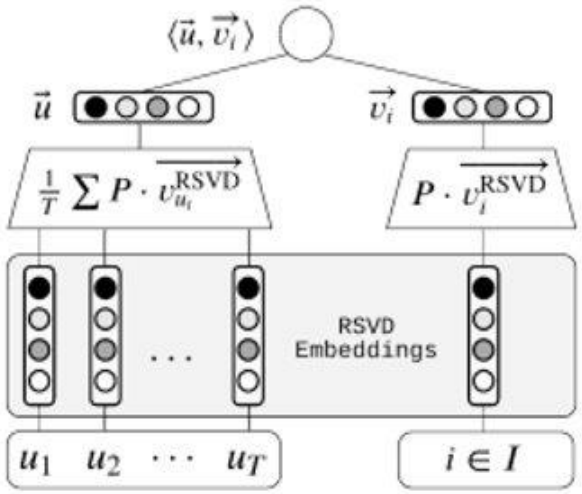
General framework

We are interested in decision rules of the form: $\boldsymbol{score}(\boldsymbol{x}, \boldsymbol{a}) \sim f_{\Xi}(\boldsymbol{x})^T h_{\beta}(\boldsymbol{a})$

Why ?

- Learn from vast amounts of heterogeneous sources of data, side information …
- Enable transfer between tasks

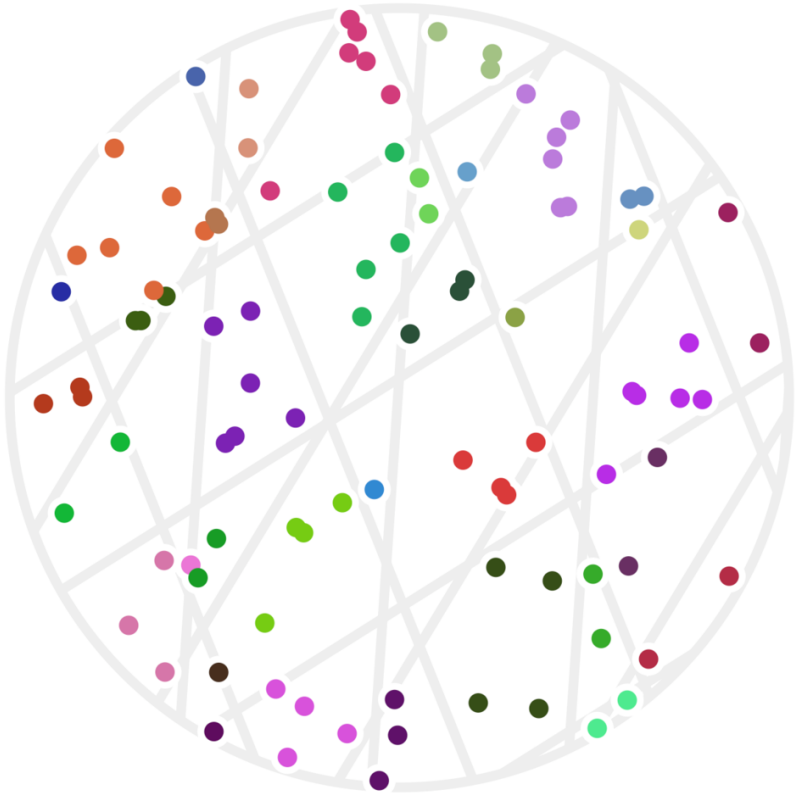
General framework



The now ubiquitous Two Tower style model.
(Criteo DeepKNN, Facebook DLRM …)

You can make the towers as shallow or deep as you want/need #deeplearning

Maximum Inner Product Search



$$\text{argsort}(f_{\Xi}(\boldsymbol{x})^T \boldsymbol{\beta})$$

$f_{\Xi}(\boldsymbol{x}) - D$
$\boldsymbol{\beta} - D \times P$

Sorting is $O(P \log P)$
Top-K is $O(P)$

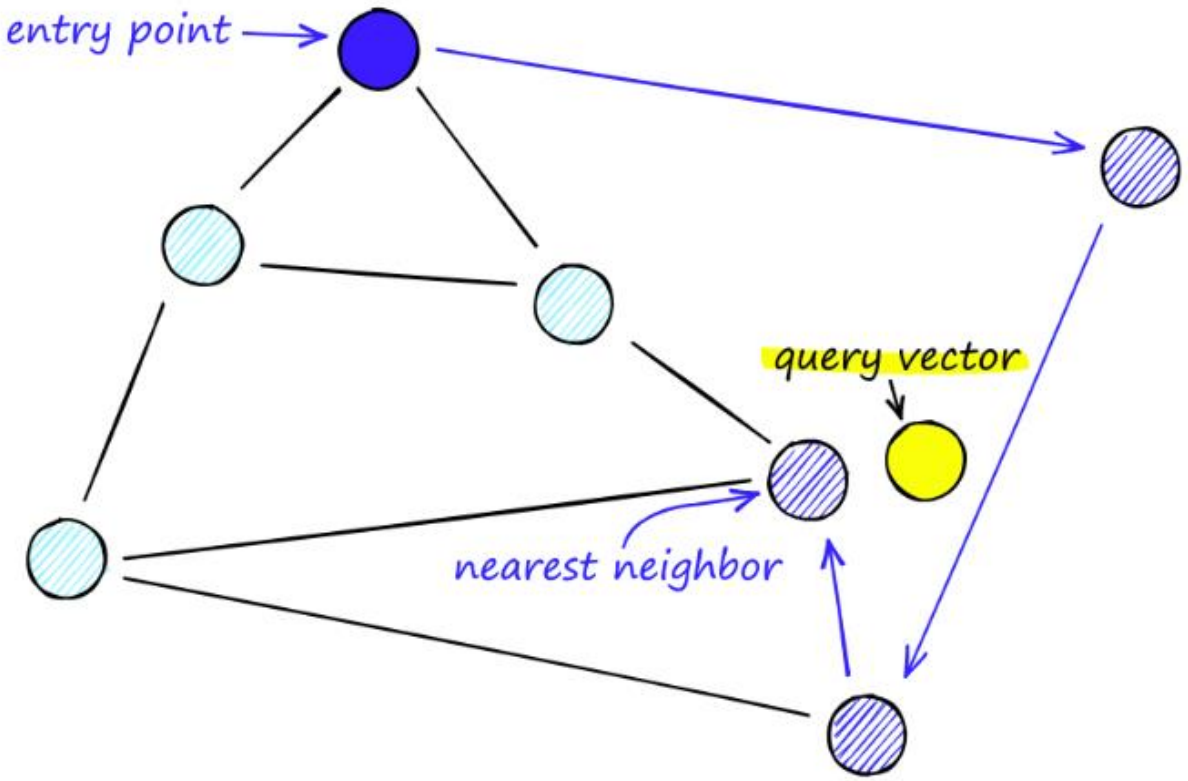
Approximate algorithms that "look like" $O(\log P)$

e.g. Navigable Small World

Side note: Hierarchical Navigable Small Worlds (HSNW)

General recipe for Graph Based NN methods

- Build a graph where close vectors are linked together
- At query time, route your query vector through the graph greedily, navigating to the closest neighbour at each time
- Stop when you hit a local minimum

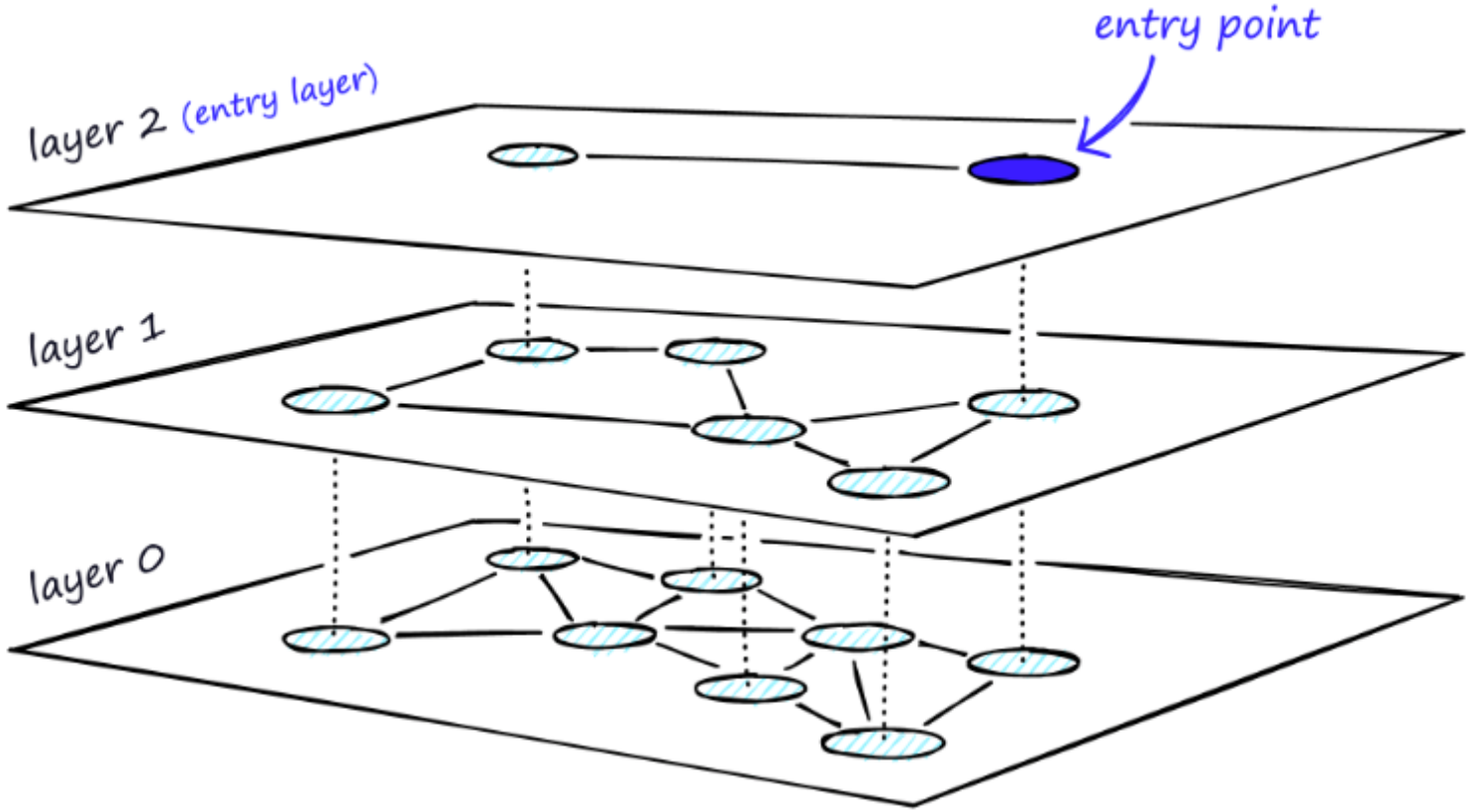


Credits: https://www.pinecone.io/learn/hnsw/

Side note: Hierarchical Navigable Small Worlds (HSNW)

The hierarchical part

- We start by a “zoomed ou” view of the neighborhood graph
- And we zoom in after routing at each successive layer
- More fine grained search as we go deeper

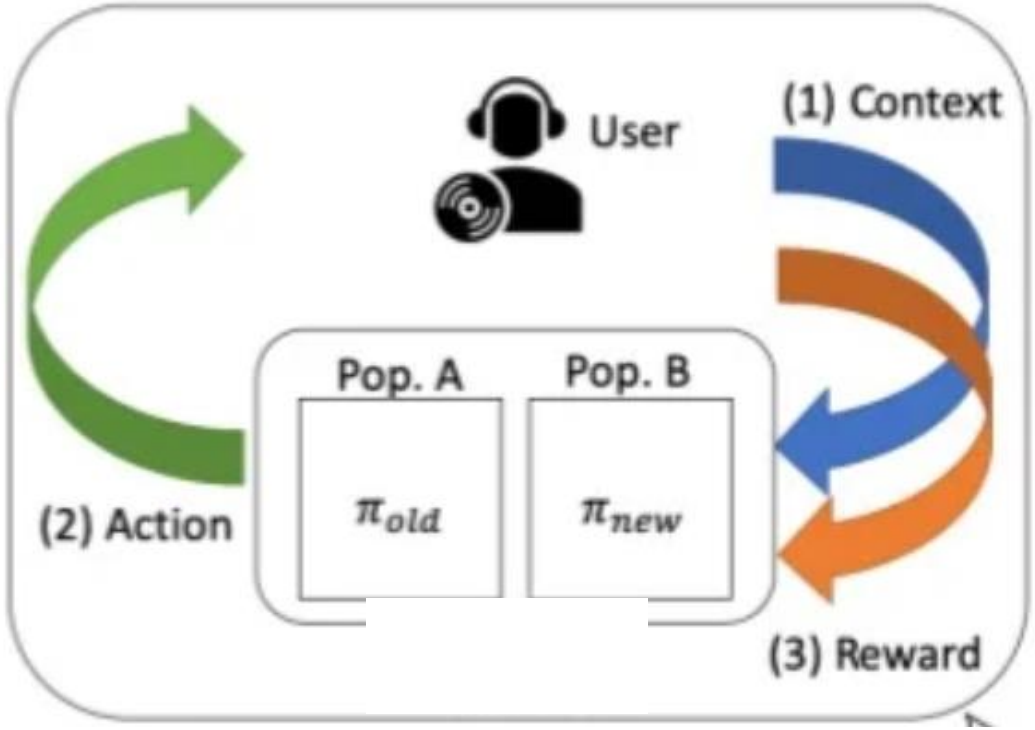


Credits: https://www.pinecone.io/learn/hnsw/

Introduction – The Science

Reward Optimizing Recommendation

Reward Optimizing Recommendation



Population A

Product view

Recommend slate

slate = A()

Product view

Recommend slate

slate = A( , )

No Sale

Population B

Product view

Product view

Recommend slate

slate = B( , )

Product view

Recommend slate

slate = B( ,  , )

Sale

Which is better A() or B()?

A() and B() are mappings from lists of items to lists of items e.g.

= B( , )

A() and B() may be constrained by engineering limitations

An A/B test can measure the timeline reward accurately (If we make the stable unit treatment value assumption).

How do we do this offline?

Product view

Product view

Recommend slate

Product view

Recommend slate

Sale

How do we do this offline?

Product view

Product view

Recommend slate

Product view

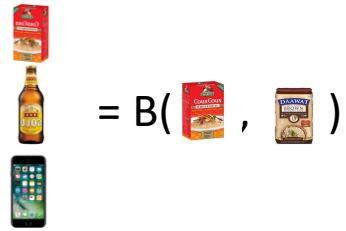
Recommend slate

Sale

This is really difficult to do!

Non-Reward Optimizing Recommendation with Pseudo Rewards

Our problem is that our recommender is a mapping:



And we're trying to answer the counterfactual question "What would happen if I had a different mapping ?" from data generated by a previous mapping, where rewards could be delayed, …

Product view

Product view

Product view

Product view

Product view

Product view

Becomes a positive example

Product view

Product view

Product view

$$L(\Xi, \boldsymbol{\beta}) = \log \sigma\{f_{\Xi}(v_1 = \text{  }, v_2 = \text{  })^T \boldsymbol{\beta_{a^+}} - f_{\Xi}(v_1 = \text{  }, v_2 = \text{  })^T \boldsymbol{\beta_{a^-}}\}$$

Becomes a positive example

Product view

Product view

Product view

$$L(\Xi, \boldsymbol{\beta}) = \log \sigma\{f_{\Xi}(v_1 = \square, v_2 = \square)^T \boldsymbol{\beta}_{a^+} - f_{\Xi}(v_1 = \square, v_2 = \square)^T \boldsymbol{\beta}_{a^-}\}$$

Becomes a positive example

The negative might be sampled uniformly or using popularity

How do we use recommendation feedback ?

First answer, use a ranking loss function while training on historical recommendations and feedback.

e.g. Some pairwise ranking loss such as BPR

Product view

Product view

Product view

Product view

Product view

Recommend Slate

Product view

Product view

Recommend Slate

No Click

Click

No Click

Product view

Product view

Recommend Slate

No Click

Click

No Click

$$L(\Xi, \boldsymbol{\beta}) = \log \sigma\{f_{\Xi}(v_1 = \text{  }, v_2 = \text{  })^T \boldsymbol{\beta}_{a^+} - f_{\Xi}(v_1 = \text{  }, v_2 = \text{  })^T \boldsymbol{\beta}_{a^-}\}$$

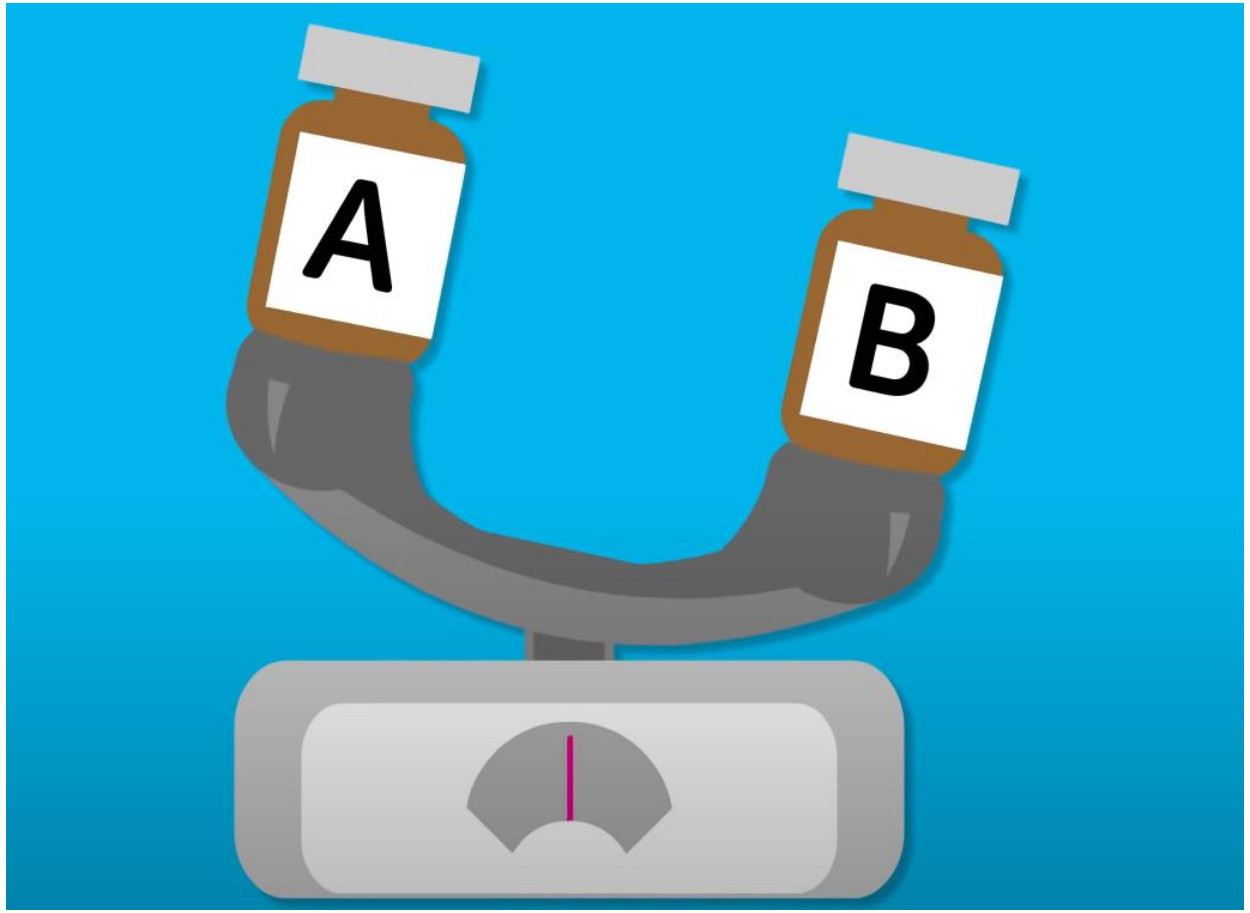
Notebook 1

https://colab.research.google.com/github/otmhi/Reward-Optimizing-Reco/blob/main/Reward_Optimizing_Slate_Recommendation_with_DL_and_MIPS_Part_1.ipynb

Reward on the Timeline

Neither Evidence Based Medicine nor Reinforcement Learning offer easy off the shelf solutions

Randomized Control Trial in Medicine



- **In medicine a randomized control trial, splits the population into two groups:**

- **We have:**

- **A potentially strong intervention**
- **Little personalization (clear membership)**
- **Usually just two actions**
- **Lots of negative results!**

Product
View

Product View

Product View

Product View

Product View

Recommend Slate

Product View

Product View

Recommend Slate

Slate =

$$\text{argsort}(f(v_1 = \text{  } \; v_2 = \text{  })^T \boldsymbol{\beta})_{1:3}$$

Product View

Product View

Recommend Slate

No click

Slate =

$$\text{argsort}(f(v_1 = \boxed{}\ v_2 = \boxed{})^T \boldsymbol{\beta})_{1:3}$$

Product View

Product View

Recommend Slate

Product View

Slate =

$$\text{argsort}(f(v_1 = \text{  } \quad v_2 = \text{  })^T \boldsymbol{\beta})_{1:3}$$

Product View

Product View

Recommend Slate

Product View

Recommend Slate

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $v_3 =$  $)^T \boldsymbol{\beta})_{1:2}$

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $)^T \boldsymbol{\beta})_{1:3}$

Product View

Product View

Recommend Slate

Product View

Recommend Slate

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $v_3 =$  $)^T \boldsymbol{\beta})_{1:2}$

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $)^T \boldsymbol{\beta})_{1:3}$

Product View

Product View

Recommend Slate

Product View

Recommend Slate

No click

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $v_3 =$  $)^T \boldsymbol{\beta})_{1:2}$

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $)^T \boldsymbol{\beta})_{1:3}$

Product View

Product View

Recommend Slate

Product View

Recommend Slate

Sale

Slate = argsort($f(v_1 =$  $v_2 =$  $v_3 =$  $)^T \boldsymbol{\beta})_{1:2}$

Slate = argsort($f(v_1 =$  $v_2 =$  $)^T \boldsymbol{\beta})_{1:3}$

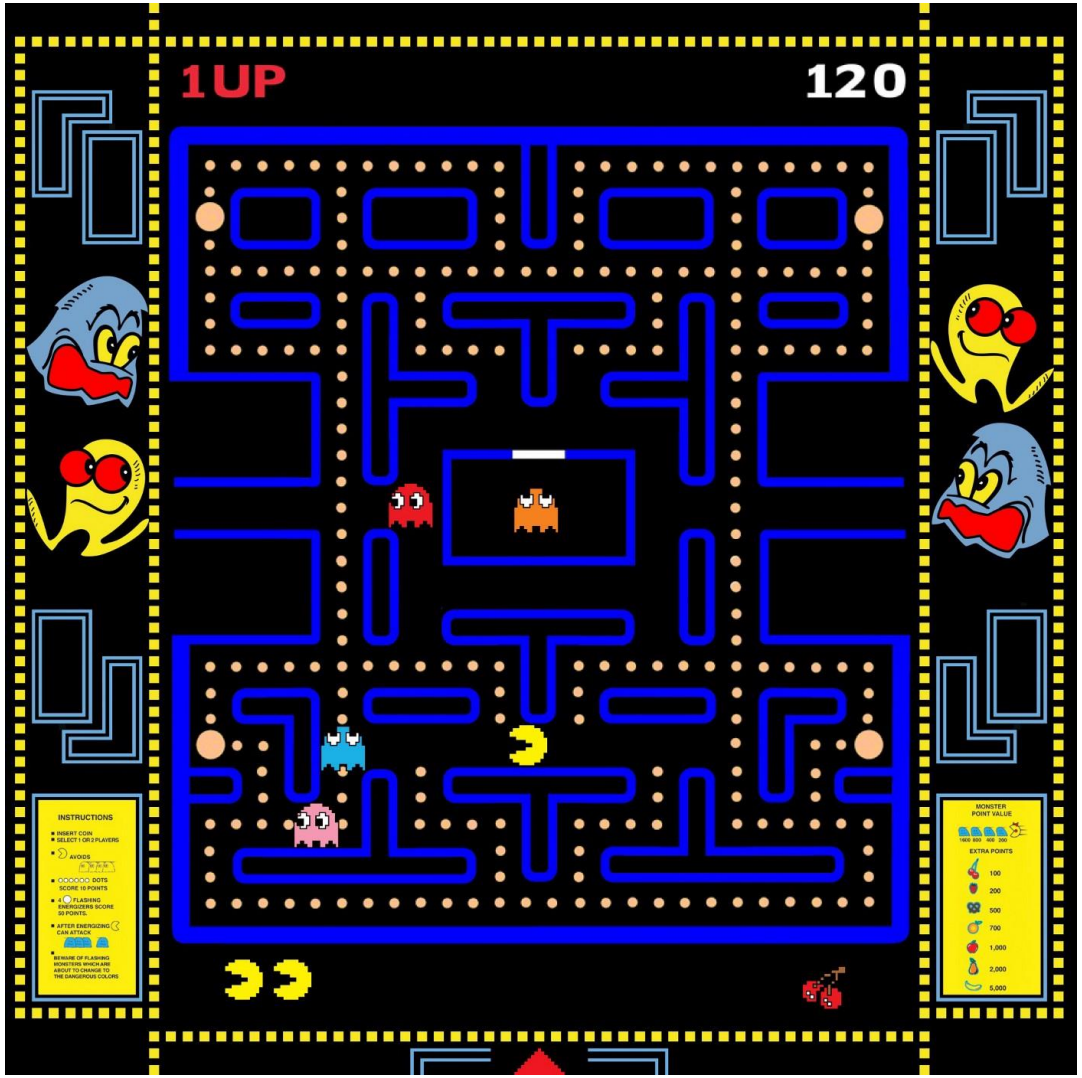
RCT in Medicine
vs
A/B Test for a Recommender System

	Medicine	**RecSys**
Personalization segments	The user is who they are. Small number of categories e.g., age band or sex.	User is what they do. If they view an item – then that is who they are and what we want to personalize on. We learn about who the user is asynchronously. The interventions / recommendations might actually cause the events that we personalize on.

RCT in Medicine
vs
A/B Test for a Recommender System

	Medicine	RecSys
Personalization segment size	Small: age bands, sex, simple medical history	Complex – everything we know.
Personalization action size	Often just intervention and placebo – (dose and dosing strategy)	With a single item slate – very big. With a large slate – the scale of the number of particles in the universe1
Effect size	Large, if it works.	Usually relatively small.

Reinforcement Learning for Game Playing



- **In certain very successful RL problems the problem is both similar and different to recommendation**
- **We have:**
  - **No cost to losing many times before you win**
  - **Strong reward signals with little noise**
  - **Sequential steps of actions are critical to unlocking reward**

Reinforcement Learning for RecSys

- **RL for RecSys**
- **In principle it is the right formulation to move reward to the timeline**
- **We have:**
- **High cost to performing poor actions in the wild**
- **Lots of noise**
- **Modest episodic features**

Decision Theoretic Recommendation

Separate the Technology and the Science of Recommendation

Decision Theory Basics

What are the **states of nature** that might occur?

Rain / No Rain

Decision Theory Basics

What are the **actions** that we might do?

Carry an umbrella / Leave umbrella

Leonard “Jimmie” Savage in 1951

State / Act	Rain	Shine
Carry	Inconvenience and wet feet	Inconvenience and slight embarrassment
Don’t carry	Miserable drenching	Bliss unalloyed

Leonard “Jimmie” Savage in 1951

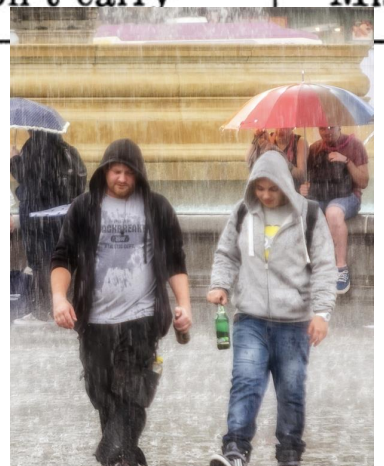
Act \ State	Rain	Shine
Carry	Inconvenience and wet feet	Inconvenience and slight embarrassment
Don’t carry	Miserable drenching	Bliss unalloyed

Leonard “Jimmie” Savage in 1951

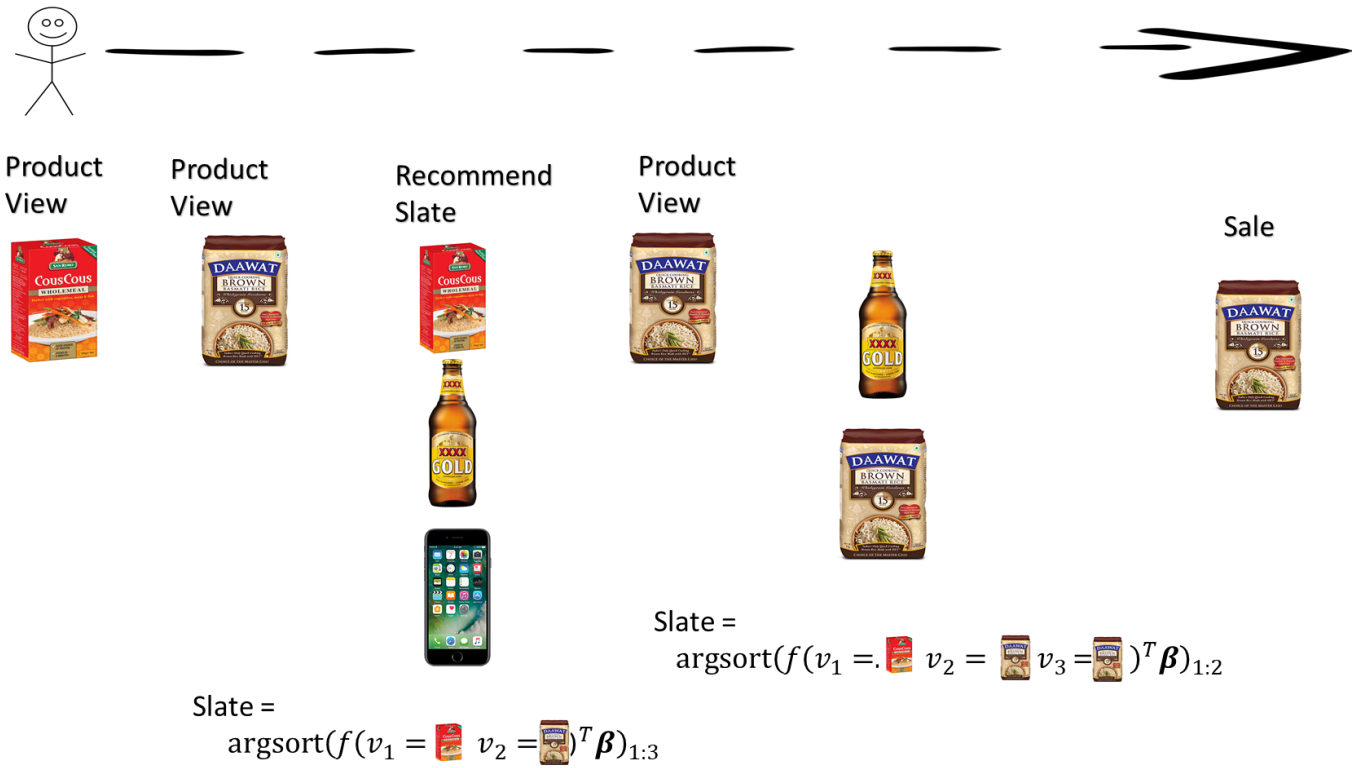
Act \ State	Rain	Shine
Carry	Inconvenience and wet feet	Inconvenience and slight embarrassment
Don’t carry	Miserable drenching	Bliss unalloyed

Leonard “Jimmie” Savage in 1951

State / Act	Rain	Shine
Carry	Inconvenience and wet feet	Inconvenience and slight embarrassment
Don’t carry	Miserable drenching	Bliss unalloyed



The model and the decision rule for RecSys

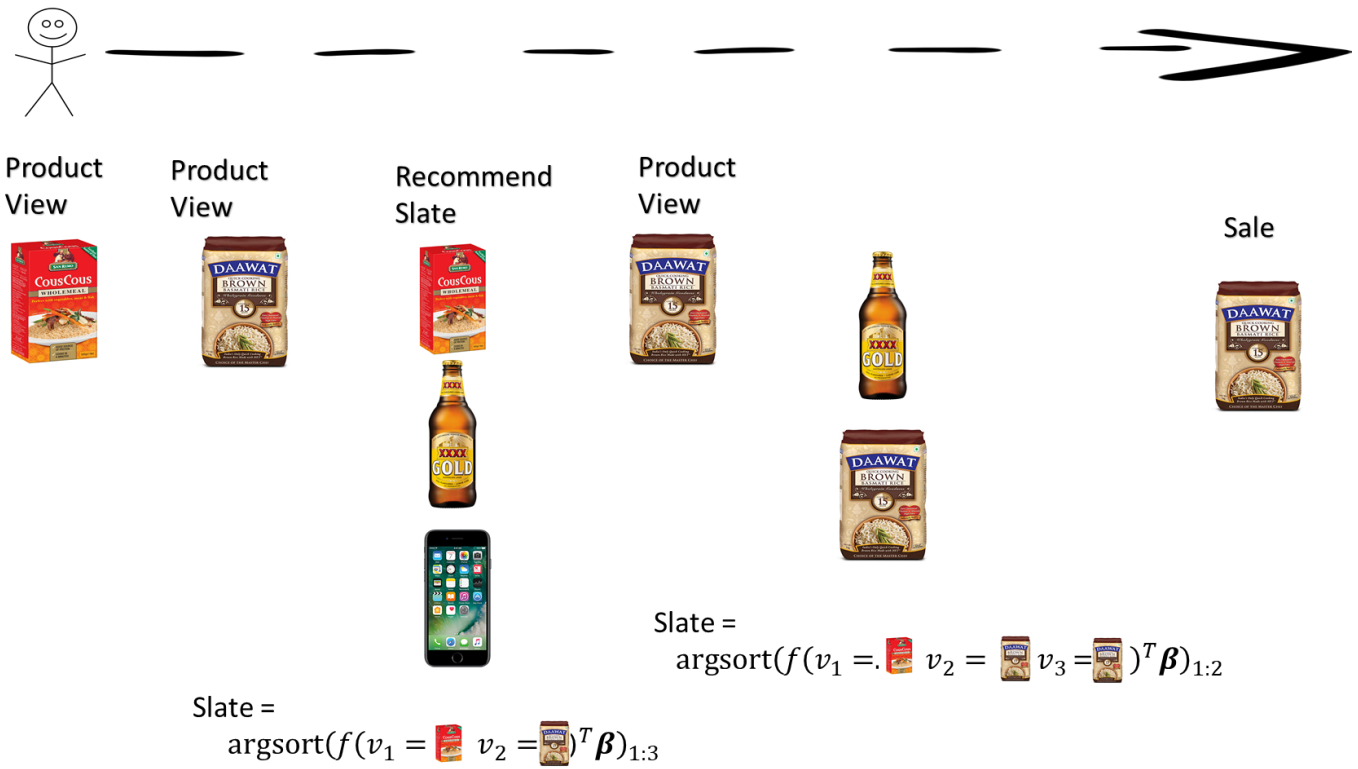


Model:

$P($ 

$)$

The model and the decision rule for RecSys



The model and the decision rule with the contextual bandit assumption:

Reward function (model):

$$R(a, \boldsymbol{x})$$

Marginal context (model)

$$P(\boldsymbol{x})$$

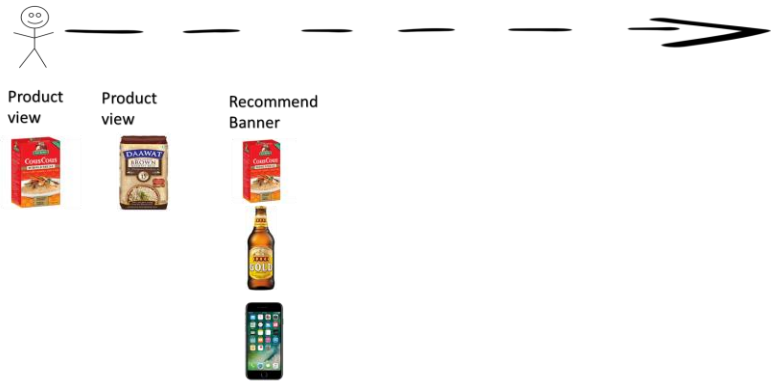
Decision Rule

$$\text{argsort}(f(\boldsymbol{x})^T\boldsymbol{\beta})_{1:K}$$

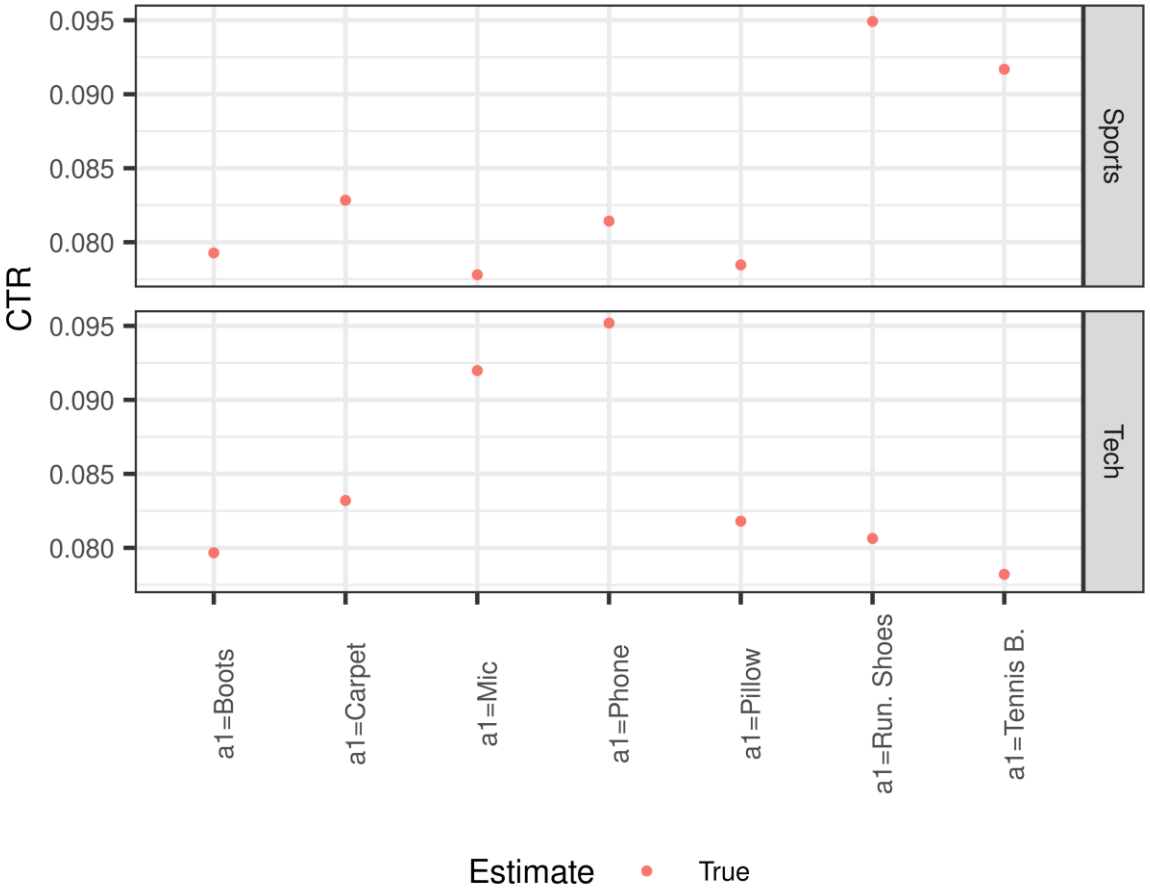
“Science”: Model based reward estimators

Estimate the reward of a recommendation using a model

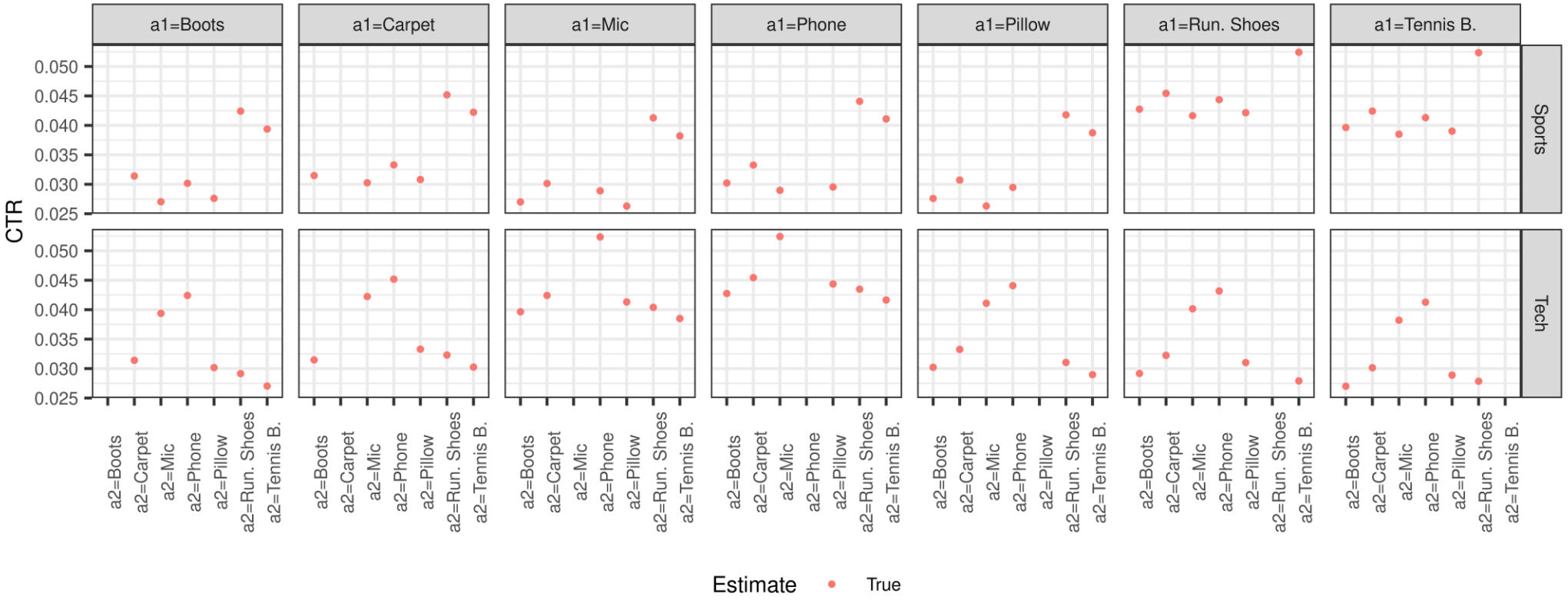
Contextual Bandit Assumption



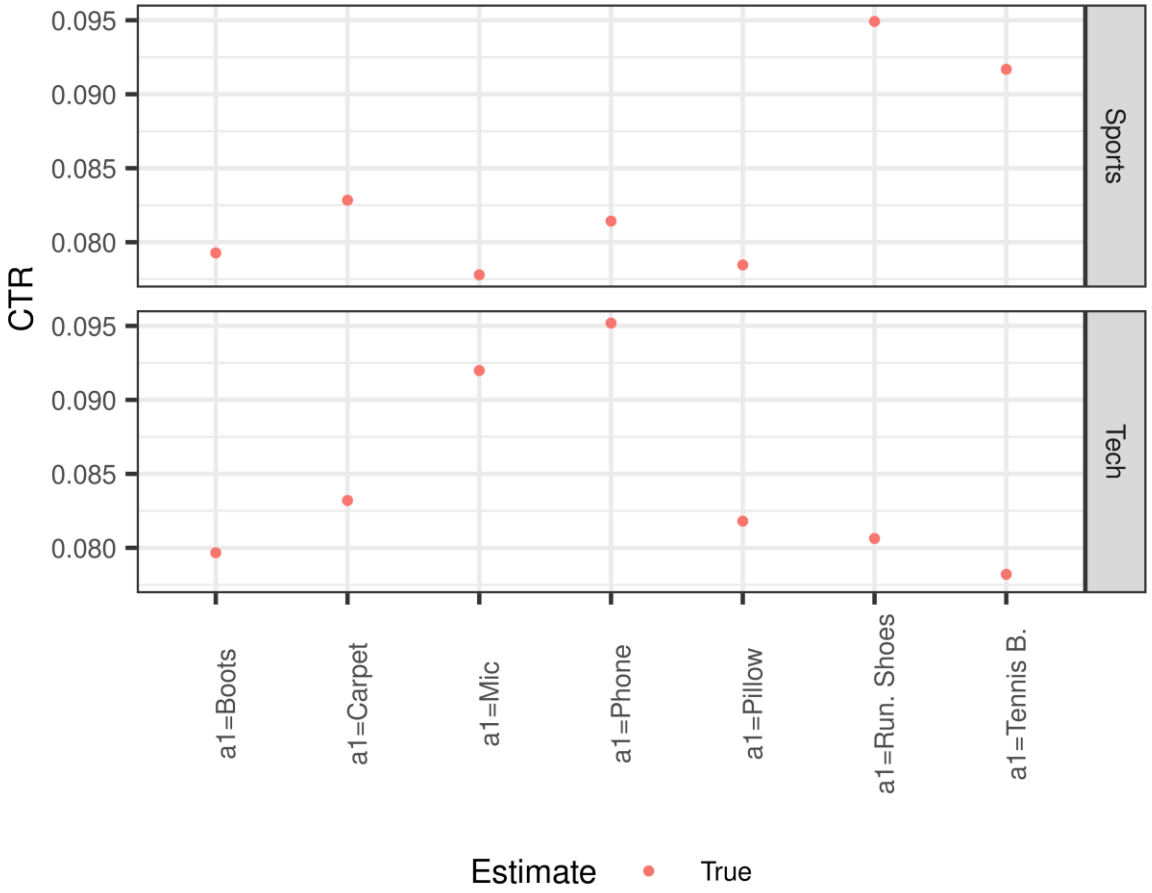
Oracle for a slate of size 1



Oracle for a slate of size 2

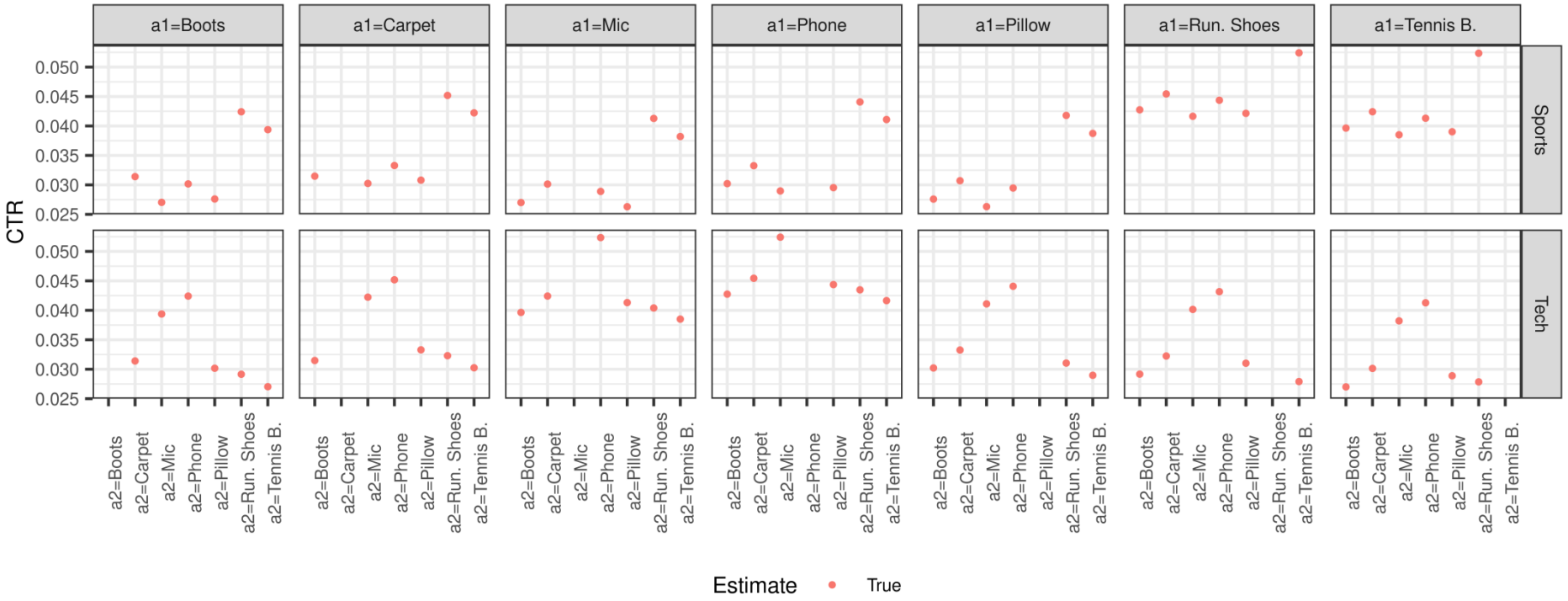


Oracle for a slate of size 1 – “Science”



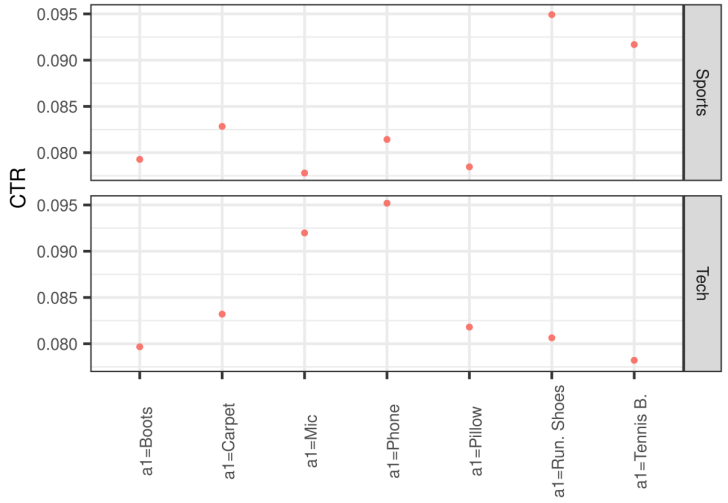
Estimate the CTRs everywhere with a slate of size 1

Oracle for a slate of size 2 – “Science”



Estimate the CTRs everywhere with a slate of size 2

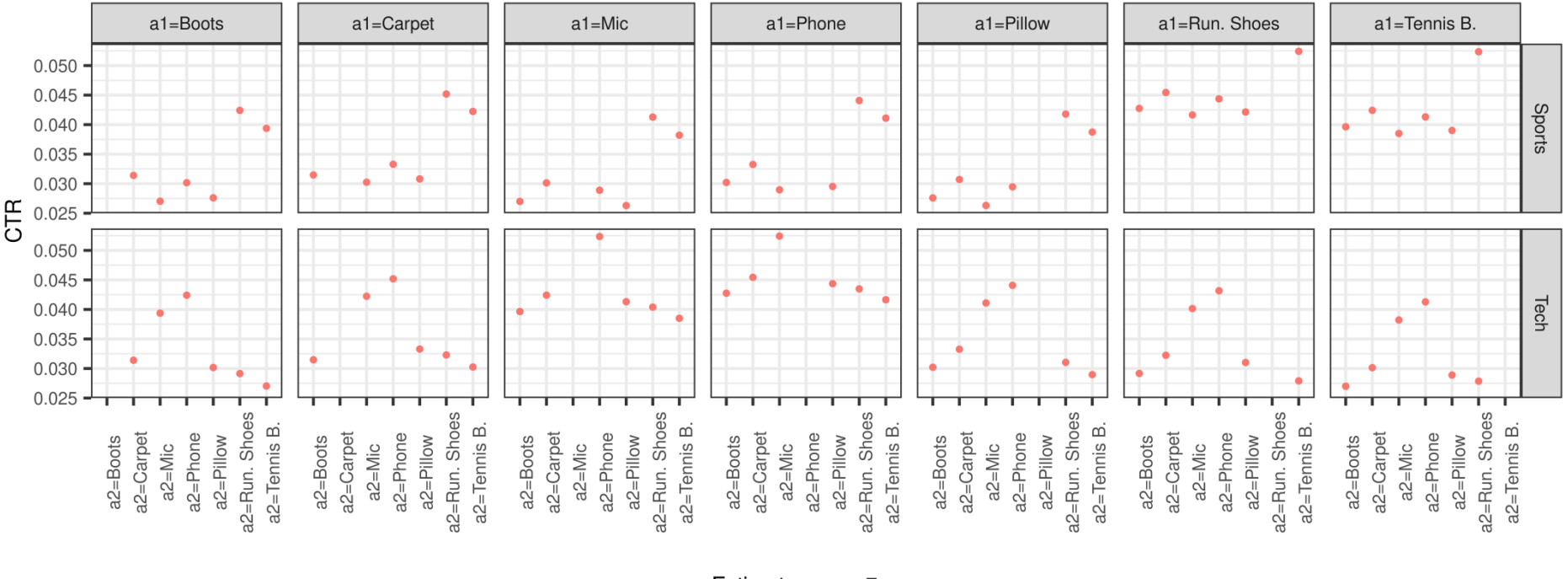
Oracle – “Science”



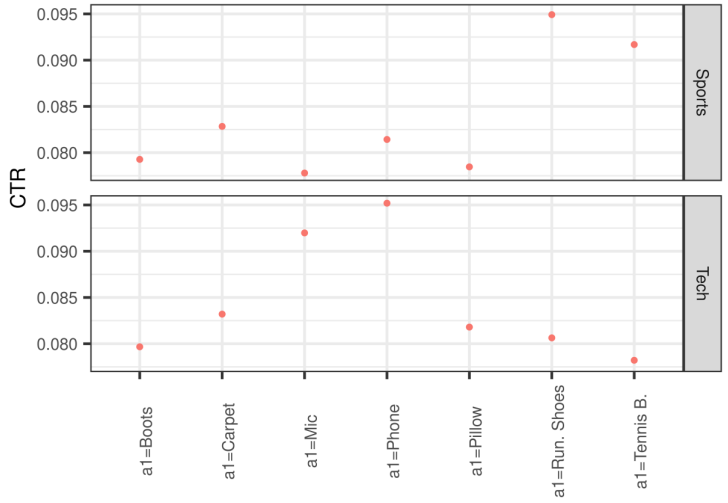
This is the reward function $R(a_1, \boldsymbol{x})$ and $R(a_1, a_2, \boldsymbol{x})$

… but we need to estimate $R(\cdot)$ from the recommender system log.

If we have logs of having tried all the recommendations with all the contexts – we can produce an estimate.

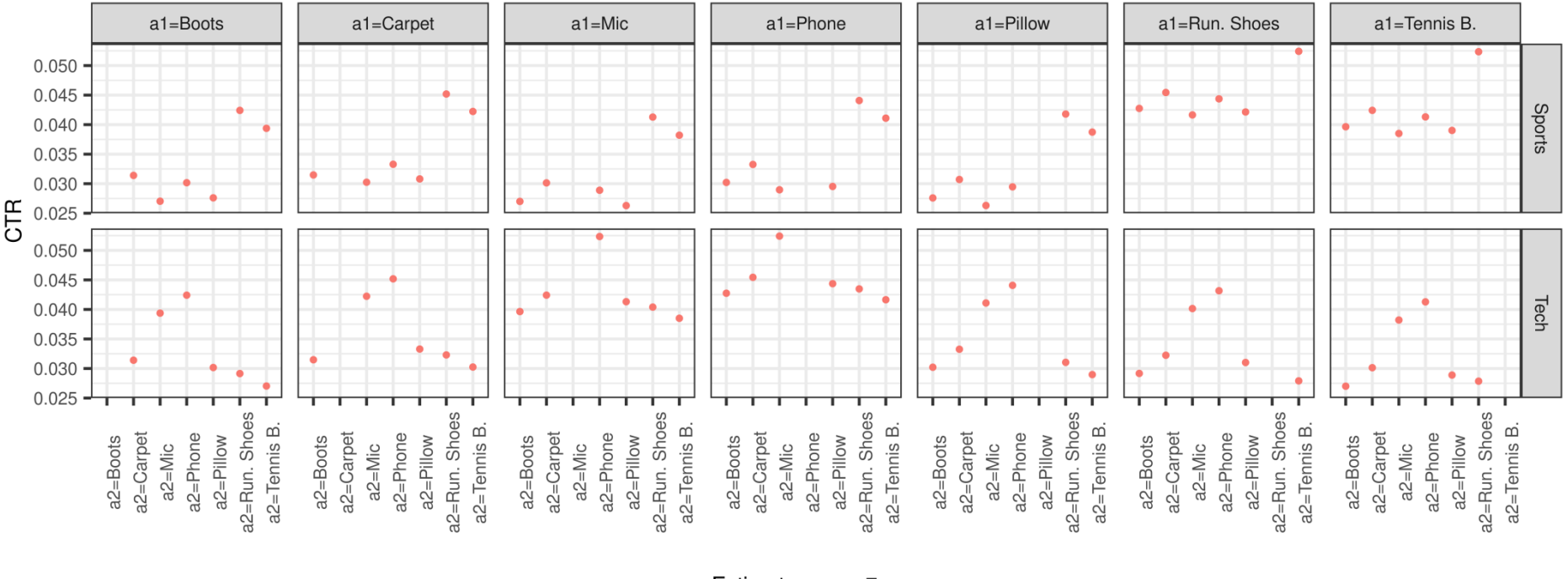


Oracle – “Science”

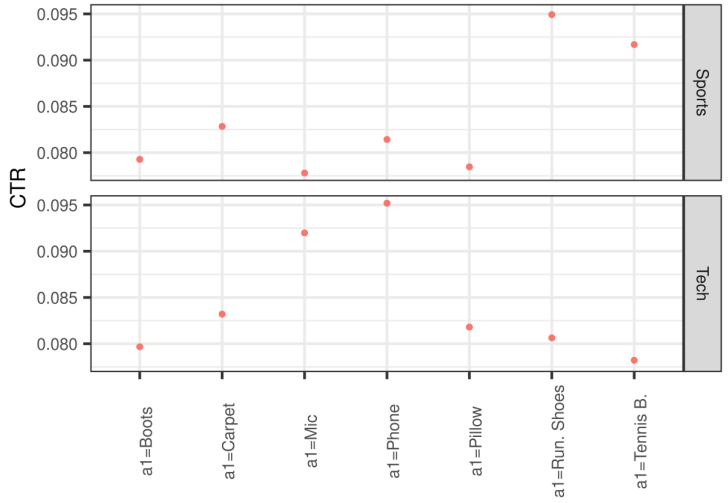


Three ways to estimate $R(\cdot)$:

- Apply directly to the real world
- Reward model
- Horvitz-Thompson Estimator

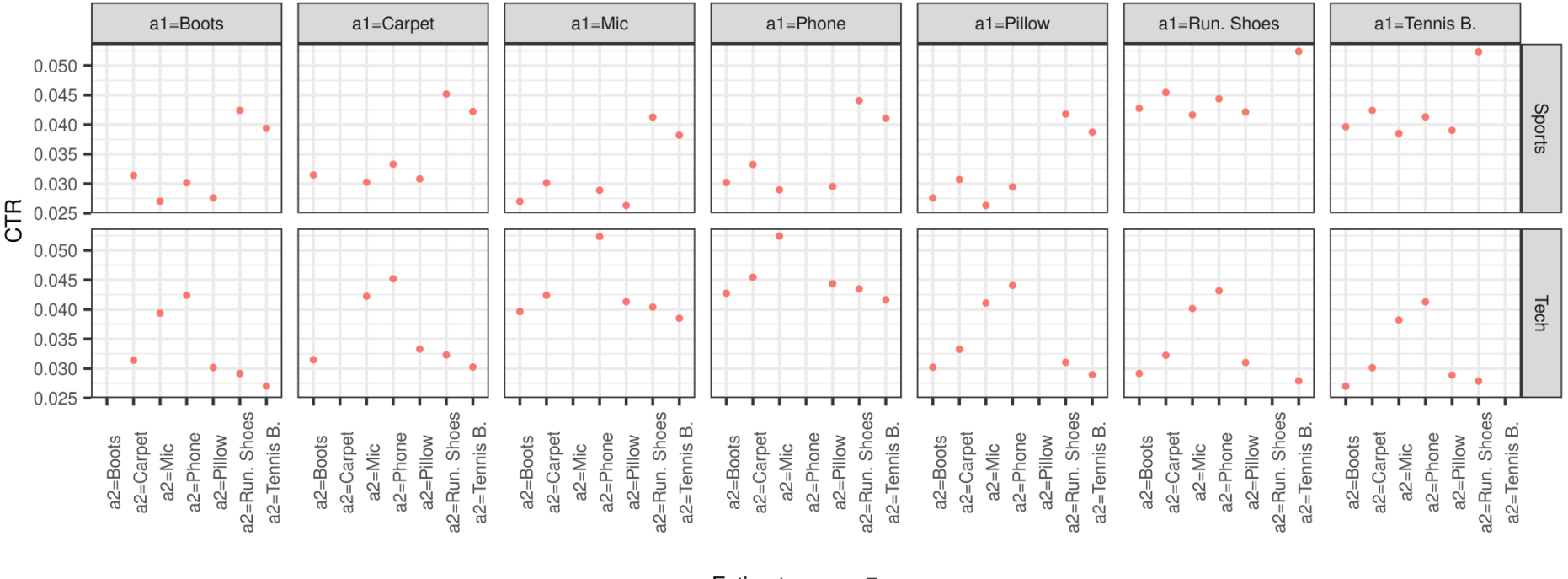


Oracle – “Technology”



Run shoes = $\text{argmax}(f_\Xi(\text{Sports})^T \boldsymbol{\beta})$

Mic = $\text{argmax}(f_\Xi(\text{Tech})^T \boldsymbol{\beta})$



Run shoes, Ten Balls = $\text{argsort}(f_\Xi(\text{Sports})^T \boldsymbol{\beta})_{1:2}$

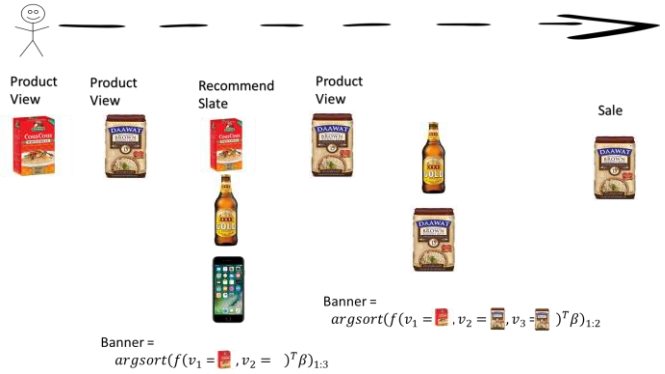
Mic, Phone = $\text{argsort}(f_\Xi(\text{Tech})^T \boldsymbol{\beta})_{1:2}$

Single Item Recommendations

Just do it!

Just do it!

One way to get an unbiased estimate of $R(a, \mathbf{x})$ is just to do it!



$$a_1, a_2 \sim \pi(a_1, a_2|\mathbf{z})$$

- Unbiased
- You do not need to make the contextual bandit assumption

- You might go broke before you converge
- Noisy

Single Item Recommendations

Model Based Reward Estimation

Logistic Regression Reward Model (e.g. BLOB)

$$R(a, \boldsymbol{x}) = P(c = 1|a, \boldsymbol{x}) = \sigma(g_{\Gamma}(\boldsymbol{x})^T \boldsymbol{\Psi}_{\boldsymbol{a}})$$

BLOB – used priors to accelerate learning the reward signal

Because of the parameterization of the reward model, we automatically get a MIPS:

$$a^* = \text{argmax}_a(f_{\Xi}(\boldsymbol{z})^T \boldsymbol{\beta})$$

with $f_{\Xi}(\boldsymbol{x}) = g_{\Gamma}(\boldsymbol{x})$ and $\boldsymbol{\beta} = \boldsymbol{\Psi}$

Logistic Regression Reward Model (e.g. BLOB)

$$R(a, \boldsymbol{x}) = P(c = 1 | a, \boldsymbol{x}) = \sigma(g_{\Gamma}(\boldsymbol{x})^T \boldsymbol{\Psi})$$

BLOB – used priors to accelerate learning the reward signal

Because of the parameterization of the reward model, we automatically get a MIPS:

$$a^* = \text{argmax}_a (f_{\Xi}(\boldsymbol{z})^T \boldsymbol{\beta})$$

with $f_{\Xi}(\boldsymbol{x}) = g_{\Gamma}(\boldsymbol{x})$ and $\boldsymbol{\beta} = \boldsymbol{\Psi}$

This is sufficient, but not necessary to build the MIPS, and it can result in larger than necessary embeddings

Logistic Regression Reward Model (e.g. BLOB)

$$R(a, \boldsymbol{x}) = P(c = 1|a, \boldsymbol{x}) = \sigma(g_{\Gamma}(\boldsymbol{x})^T \boldsymbol{\Psi})$$

BLOB – used priors to accelerate learning the reward signal

Because of the parameterization of the reward model, we automatically get a MIPS:

$$a^* = \text{argmax}_a(f_{\Xi}(\boldsymbol{z})^T \boldsymbol{\beta})$$

with $f_{\Xi}(\boldsymbol{x}) = g_{\Gamma}(\boldsymbol{x})$ and $\boldsymbol{\beta} = \boldsymbol{\Psi}$ ← We will discuss alternative ways to get $f_{\Xi}(\cdot), \boldsymbol{\beta}$

Logistic Regression Reward Model (e.g. BLOB)

We can split $\boldsymbol{x} = [\boldsymbol{y}, \boldsymbol{z}]$

Where $\boldsymbol{y}$ – User Engagement, and $\boldsymbol{z}$ – User interest

$$R(a, x) = P(c = 1 | a, \boldsymbol{x}) = \sigma(\boldsymbol{y}^T \boldsymbol{\phi} + g_\Gamma(\boldsymbol{z})^T \boldsymbol{\Psi})$$

$$a^* = \text{argmax}_a (f_\Xi(\boldsymbol{z})^T \boldsymbol{\beta})$$

with $f_\Xi(\boldsymbol{z}) = g_\Gamma(\boldsymbol{z})$ and $\boldsymbol{\beta} = \boldsymbol{\Psi}$

Logistic Regression Reward Model (e.g. BLOB)

We can split $\boldsymbol{x} = [\boldsymbol{y}, \boldsymbol{z}]$

Where y – User Engagement, and z – User interest

$$R(a, x) = P(c = 1|a, \boldsymbol{x}) = \sigma(\boldsymbol{y}^T \boldsymbol{\phi} + g_\Gamma(\boldsymbol{z})^T \boldsymbol{\Psi})$$

$$a^* = \text{argmax}_a(f_\Xi(\boldsymbol{z})^T \boldsymbol{\beta})$$

Nuisance parameters

with $f_\Xi(\boldsymbol{z}) = g_\Gamma(\boldsymbol{z})$ and $\boldsymbol{\beta} = \boldsymbol{\Psi}$

Product
View

Product View

Product View

Product View

Product View

Recommend

Product View

Product View

Recommend

Click/No click

Recommend = $\text{argmax}(f(v_1 =$  $, v_2 =$  $)^T \boldsymbol{\beta})$

Product View

Product View

Recommend

Click/No click

When we build a model to predict the reward, we can try to predict the reward (click).

Build a model based on:

We can also use features like:

- How active is the user?
- How big/attractive is the recommendation?

Recommend = $\text{argmax}(f(v_1 = $  $, v_2 = $  $)^T \boldsymbol{\beta})$

Product View

Product View

Recommend

Click/No click

When we build a model to predict the reward, we can try to predict the reward (click).

Build a model based on:

This (often) dominates in practice →

We can also use features like:

- **How active is the user?**
- **How big/attractive is the recommendation?**

Recommend = $\text{argmax}(f(v_1 =$  $, v_2 =$  $)^T \boldsymbol{\beta})$

Product View

Product View

Recommend

Click/No click

When we build a model to predict the reward, we can try to predict the reward (click).

We can predict well with *only* these features →

We can also use features like:

- **How active is the user?**
- **How big/attractive is the recommendation?**

Recommend = $\text{argmax}(f(v_1 =$  $, v_2 =$  $)^T \boldsymbol{\beta})$

Single Item Recommendations

Horvitz-Thompson Reward Estimation

Conditional Horvitz-Thompson Reward Estimation

$$R(a^*, \boldsymbol{x}) = \int c\, P(c|a', \boldsymbol{x})\pi^*(a'|\boldsymbol{x})da' = \int c\, \pi_0(a'|\boldsymbol{x}) \frac{P(c|a', \boldsymbol{x})\pi^*(a'|\boldsymbol{x})}{\pi_0(a'|\boldsymbol{x})} da'$$

$\pi^*(a|x) = 1$ if $a = a^*$ and $\pi^*(a|x) = 0$, otherwise.

Horvitz-Thompson Reward Estimator

$$R(a, \mathbf{x}) = \frac{1}{N_{a,\boldsymbol{x}}} \sum_{i \in (a_i = a, x_i = x)} \frac{c_i}{\pi_0(a_i | \boldsymbol{x}_i)}$$

We compute a sum over the relevant history

Horvitz-Thompson Reward Estimator

Reward

$$R(a, \mathbf{x}) = \frac{1}{N_{a,\boldsymbol{x}}} \sum_{i \in (a_i = a, x_i = x)} \frac{c_i}{\pi_0(a_i | \boldsymbol{x}_i)}$$

Horvitz-Thompson Reward Estimator

$$R(a, \mathbf{x}) = \frac{1}{N_{a,\boldsymbol{x}}} \sum_{i \in (a_i = a, x_i = x)} \frac{c_i}{\pi_0(a_i | \boldsymbol{x}_i)}$$

Propensity score

Horvitz-Thompson Reward Estimator

$$R(a, \mathbf{x}) = \frac{1}{N_{a,\boldsymbol{x}}} \sum_{i \in (a_i = a, x_i = x)} \frac{c_i}{\pi_0(a_i | \boldsymbol{x}_i)}$$

Number of records

Horvitz-Thompson Reward Estimator of a Policy

Previously we had a mapping $A(\boldsymbol{x})$ which returned a recommendation for user x. This mapping is non-differentiable, so we introduce a "policy" $\pi(a|\boldsymbol{x})$. While $\pi(a|\boldsymbol{x})$ is mathematically a probability, but it does not represent uncertainty, it simply makes the (in this case argmax) differentiable. We then get:

$$E[U|\pi(\cdot)] = \frac{1}{N}\sum_{i}\frac{c_i\,\pi(a_i|\boldsymbol{x_i})}{\pi_0(a_i|\boldsymbol{x}_i)}$$

Horvitz-Thompson Reward Estimator of a Policy

Previously we had a mapping $A(\boldsymbol{x})$ which returned a recommendation for user x. This mapping is non-differentiable, so we introduce a "policy" $\pi(a|\boldsymbol{x})$. While $\pi(a|\boldsymbol{x})$ is mathematically a probability, but it does not represent uncertainty, it simply makes the (in this case argmax) differentiable. We then get:

$$E[U|\pi(\cdot)] = \frac{1}{N}\sum_i \frac{c_i\ \pi(a_i|\boldsymbol{x_i})}{\pi_0(a_i|\boldsymbol{x}_i)}$$

If $c, a, \boldsymbol{x} \sim P(c|a,\boldsymbol{x})\pi_0(a|\boldsymbol{x})P(\boldsymbol{x})$

Then this can be viewed as an importance sampling formula for evaluating: $\int c\ P(c|a,\boldsymbol{x})\pi(a|\boldsymbol{x})P(\boldsymbol{x})da\ d\boldsymbol{x}$

Horvitz-Thompson Reward Estimator of a Policy

Previously we had a mapping $A(\boldsymbol{x})$ which returned a recommendation for user x. This mapping is non-differentiable, so we introduce a "policy" $\pi(a|\boldsymbol{x})$. While $\pi(a|\boldsymbol{x})$ is mathematically a probability, but it does not represent uncertainty, it simply makes the (in this case argmax) differentiable. We then get:

$$E[U|\pi(\cdot)] = \frac{1}{N}\sum_i \frac{c_i\,\pi(a_i|\boldsymbol{x_i})}{\pi_0(a_i|\boldsymbol{x}_i)}$$

If you run the epsilon greedy policy for a non-typical action $\pi_0(a|\boldsymbol{x}) = \frac{\epsilon}{P-1}$. This leads to massive variance..

If $c, a, \boldsymbol{x} \sim P(c|a,\boldsymbol{x})\pi_0(a|\boldsymbol{x})P(\boldsymbol{x})$

Then this can be viewed as an importance sampling formula for evaluating: $\int c\; P(c|a,\boldsymbol{x})\pi(a|\boldsymbol{x})P(\boldsymbol{x})da\; d\boldsymbol{x}$

Single Item Recommendations

Decision Rule Optimization

General Reward Estimator of a Policy

$$E[U|\pi(\cdot)] = \frac{1}{N} \sum_{i=1..N} \sum_{a'=1..P} R(a', \boldsymbol{x_n}) \pi(a'|\boldsymbol{x_n})$$

Obviously the optimal $\pi(a|\boldsymbol{x})$ has the property $\pi(a^*|\boldsymbol{x})$=1 where $a^* = \text{argmax}_a R(a, \boldsymbol{x})$

In practice we can obtain derivatives of $\pi(a|\boldsymbol{x})$ to optimize the policy/decision rule/MIPS

General Reward Estimator of a Policy

$R(a, \boldsymbol{x})$ can come from:
- a model
- Horvitz-Thompson
- Just do it …

$$E[U|\pi(\cdot)] = \frac{1}{N} \sum_{i=1..N} \sum_{a\prime=1..P} R(a', \boldsymbol{x_n}) \pi(a'|\boldsymbol{x_n})$$

Obviously the optimal $\pi(a|\boldsymbol{x})$ has the property $\pi(a^*|\boldsymbol{x})$=1 where $a^* = \text{argmax}x_a R(a, \boldsymbol{x})$

In practice we can obtain derivatives of $\pi(a|\boldsymbol{x})$ to optimize the policy/decision rule/MIPS

Optimize $\pi(a|\boldsymbol{x})$ when $R(a, \boldsymbol{x})$ is a model

If our reward model has the following parametric form:

$$R(a, \boldsymbol{x}) = P(c = 1|a, \boldsymbol{x}) = \sigma(g_\Gamma(\boldsymbol{x})^T \boldsymbol{\Psi}_a)$$

Then we already have an optimal MIPS: $f_\Xi(\boldsymbol{x})^T \boldsymbol{\beta}$, by setting

$$f_\Xi(\boldsymbol{x}) = g_\Gamma(\boldsymbol{z}), \qquad \boldsymbol{\beta} = \boldsymbol{\Psi}$$

So why would we optimize: $E[U|\pi(\cdot)] = \frac{1}{N}\sum_i R(a, \boldsymbol{x})\pi(a|\boldsymbol{x})$ to obtain a MIPS if we already have one?

Reward from a model

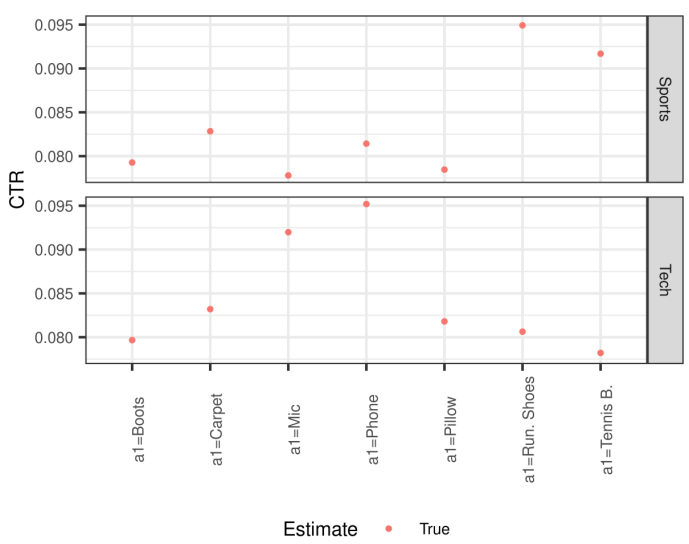
- $P(c = 1|a, \boldsymbol{x}) = \sigma(\boldsymbol{y}^T\boldsymbol{\phi} + g_\Gamma(\boldsymbol{z})^T\boldsymbol{\beta}_a)$ is sufficient but not necessary
- A few reasons we might want not to use:
  - $\sigma(\boldsymbol{y}^T\boldsymbol{\phi} + f_\Xi(\boldsymbol{z})^T\boldsymbol{\beta}_a)$ is a calibrated reward that correctly orders all actions. We only need the best action
  - Imposing the above can mean the embedding dimension D for $f_\Xi(\boldsymbol{x})^T\boldsymbol{\beta}$ might be larger than we need
  - It might be an unreasonable engineering burden to produce perfectly optimal recommendations at all times. We might want the dimension of $\boldsymbol{\beta}$ lower than $\boldsymbol{\Omega}$
  - We might prefer a more general model $P(c = 1|a, \boldsymbol{x}) = \sigma(f_\Xi(\boldsymbol{x}, a))$

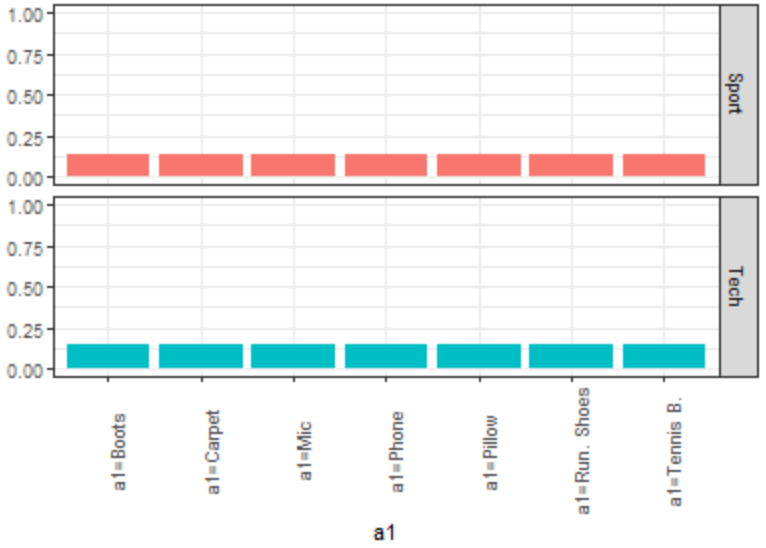
Optimizing the decision rule via a policy

$$E[U|\pi(\cdot)] = \sum_{a} R(a, \mathbf{x})\pi(a|\mathbf{x})$$

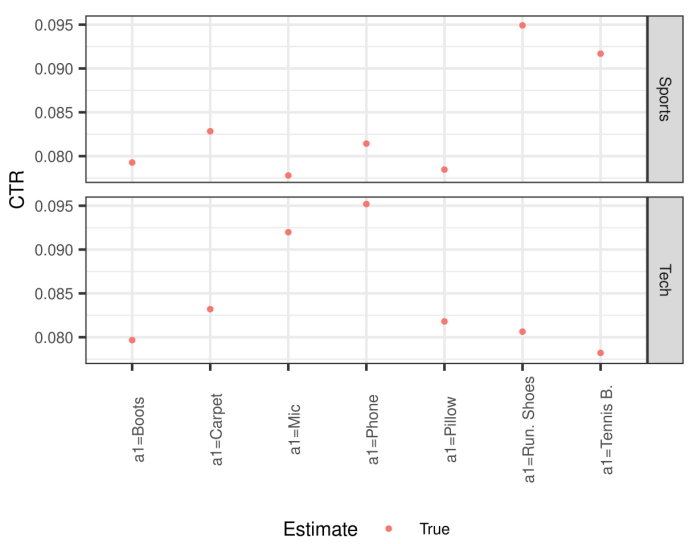
If we want $\pi(a|\boldsymbol{x})$ to be restricted to producing a MIPS we let $\pi(a|\boldsymbol{x}) = \frac{\exp(f_{\Xi}(\boldsymbol{x})^T \boldsymbol{\beta}_a)}{\sum_i \exp(f_{\Xi}(\boldsymbol{x})^T \boldsymbol{\beta}_i)}$

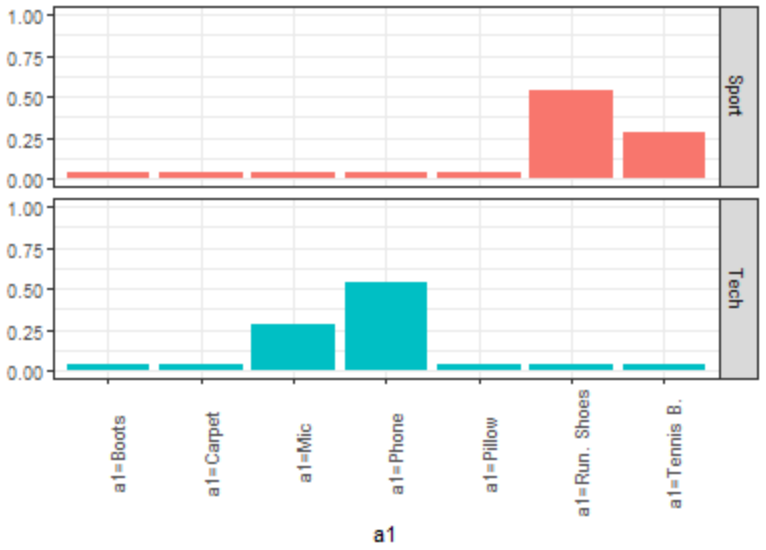
Optimizing the Policy



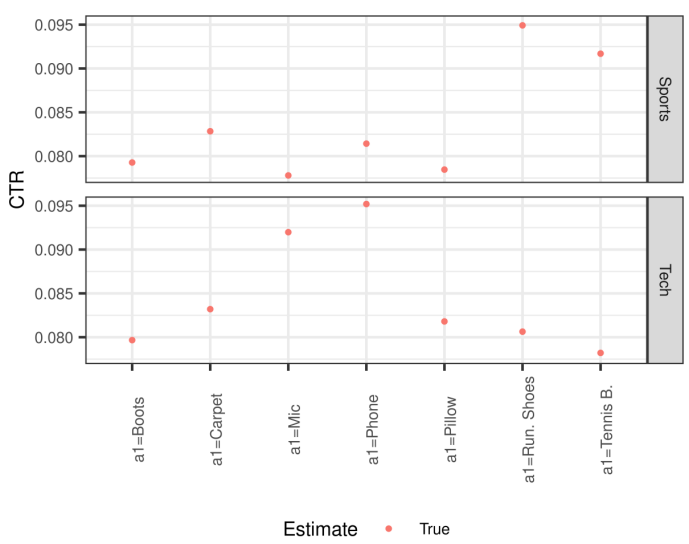


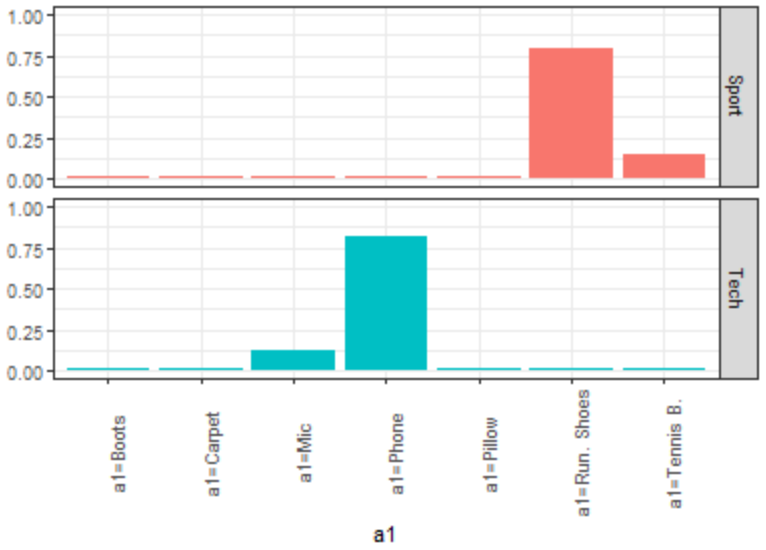
Optimizing the Policy



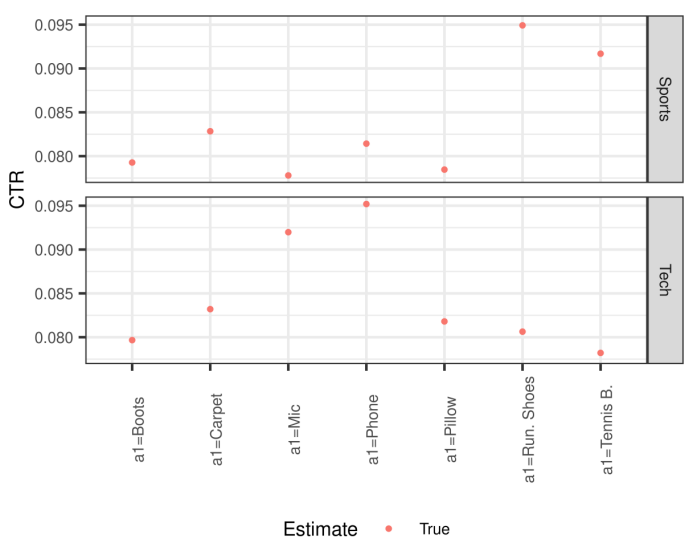


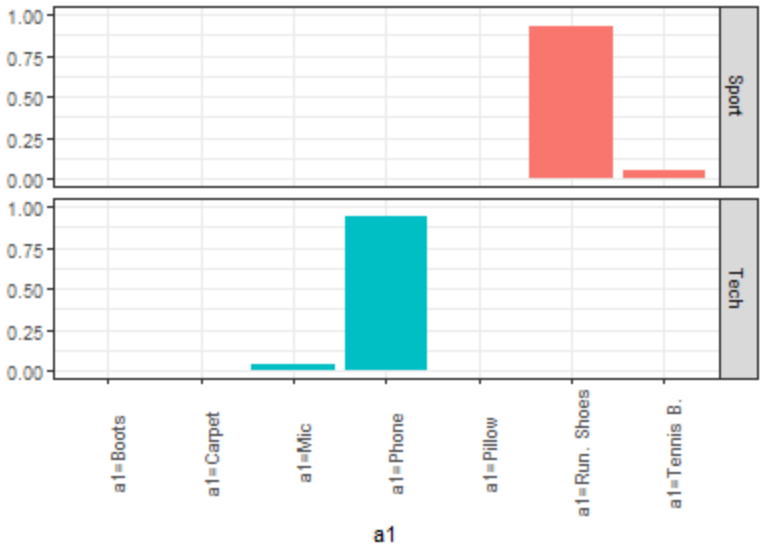
Optimizing the Policy



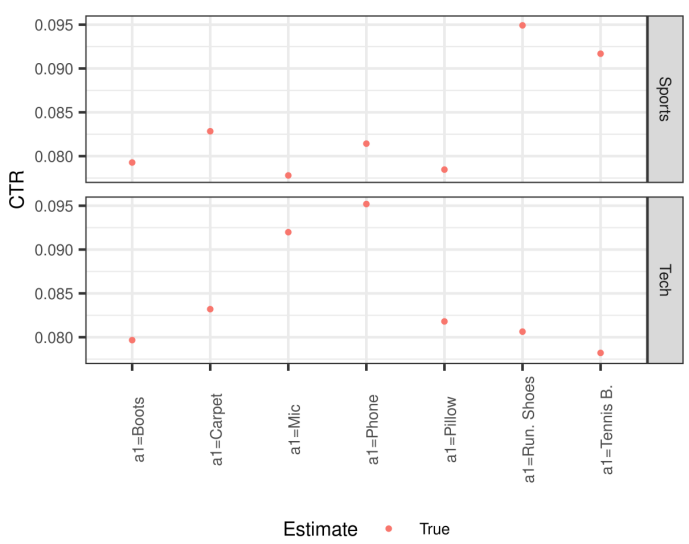


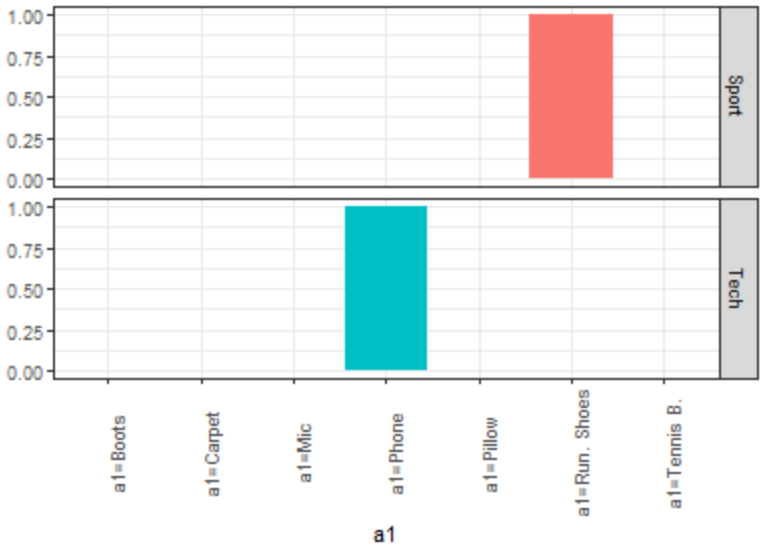
Optimizing the Policy





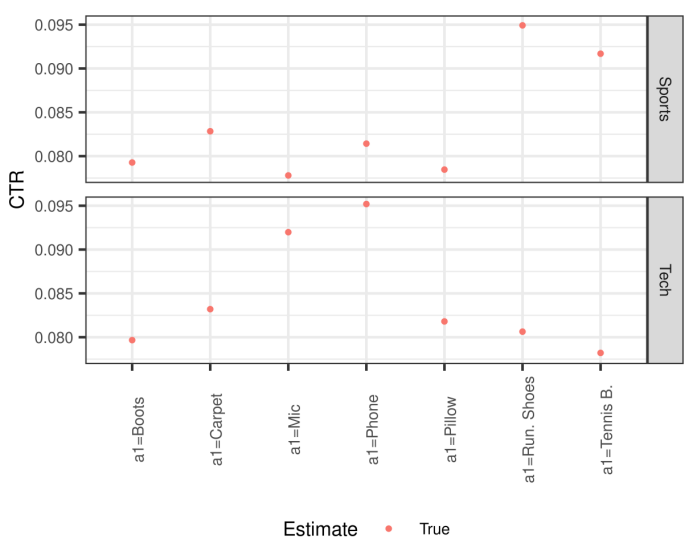
Optimizing the Policy

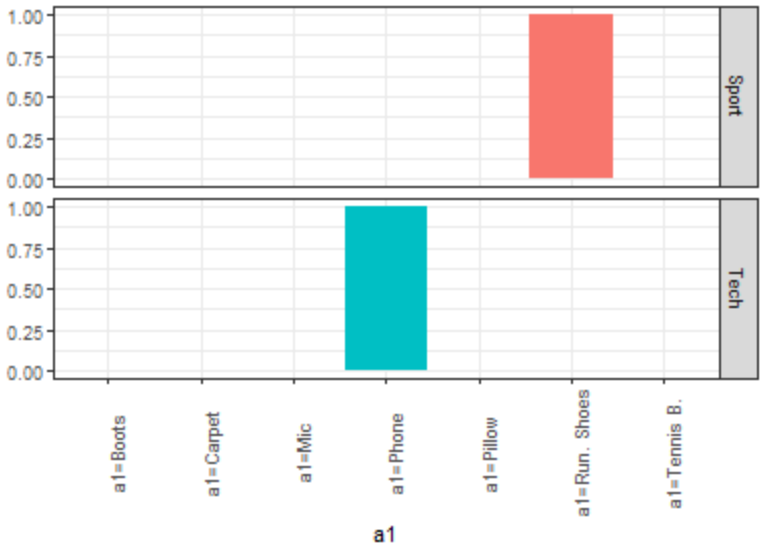




At convergence $\pi(a|\boldsymbol{x})$ is degenerate.

Optimizing the Policy

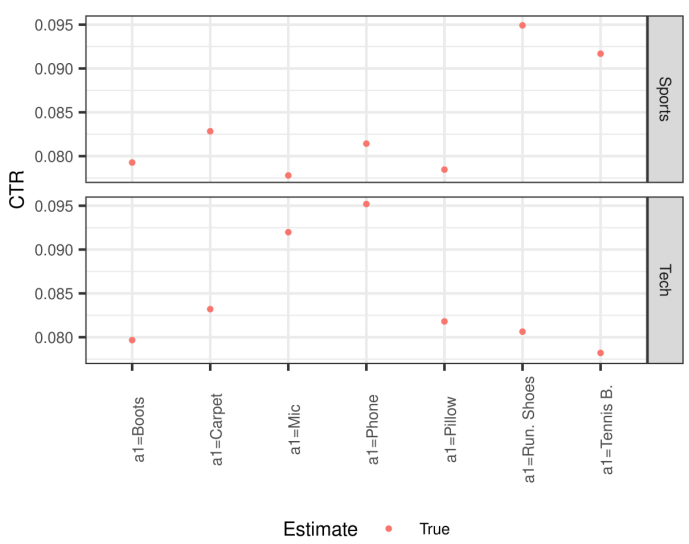


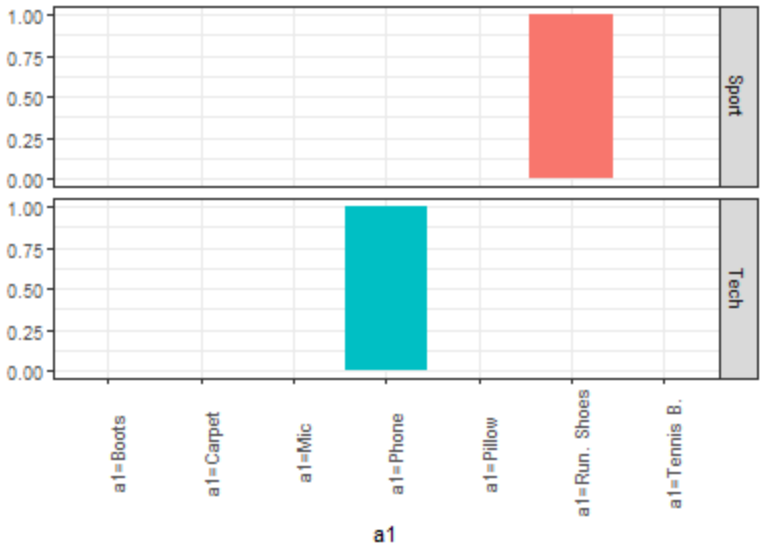


At convergence $\pi(a|\boldsymbol{x})$ is degenerate.

If the reward estimate is wrong, it will (possibly) select the wrong best action

Two parts to decision theoretic Reco





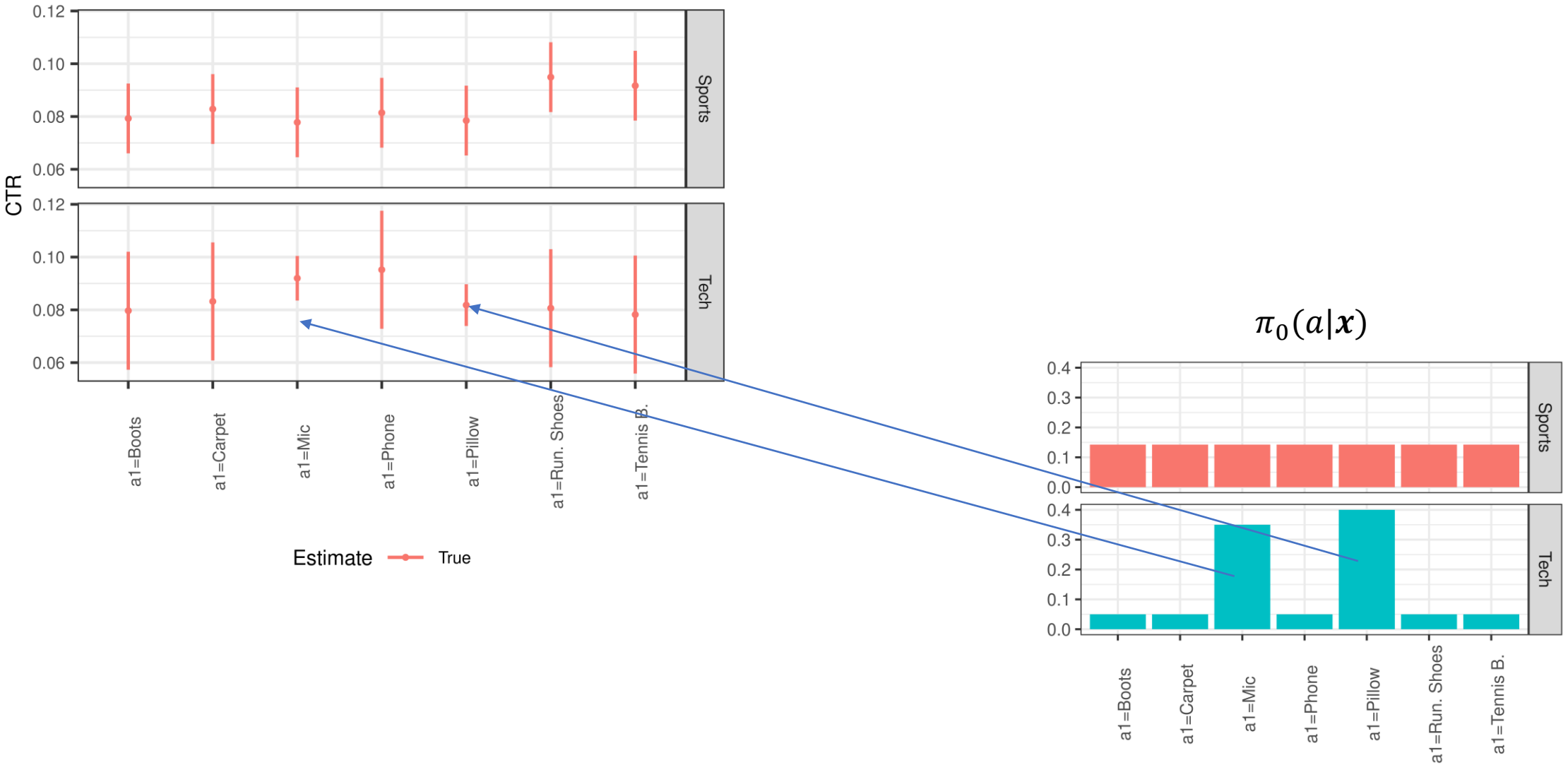
Model Based Methods
Can accelerate learning and borrow strength

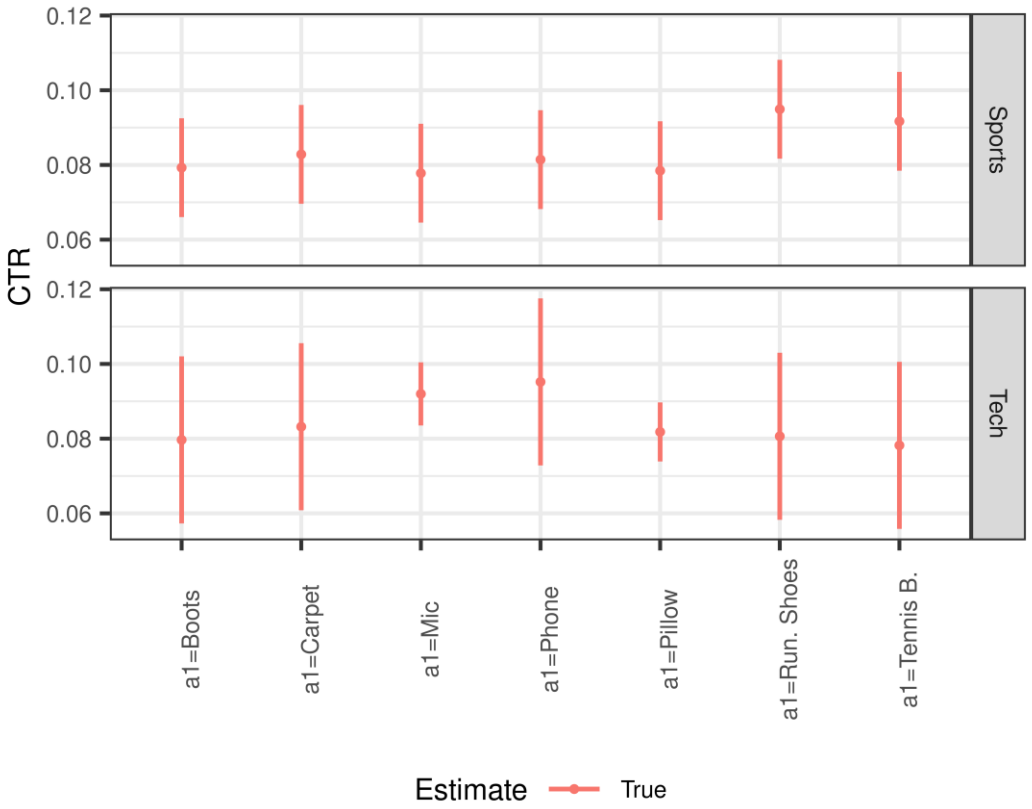
BLOB uses:
- context-context
- action-action distances
- context-action

Sakhi, O., et al. 2020

Reward estimation
- Model based
- Horvitz Thompson
- Just do it..

Decision rule optimization
- Policy learning
- Contextual bandits
- Reinforce


CTR
0.12
0.10
0.08
0.06
Sports
Tech
a1=Boots
a1=Carpet
a1=Mic
a1=Phone
a1=Pillow
a1=Run. Shoes
a1=Tennis B.
Estimate
True
$\pi_0(a|\boldsymbol{x})$
0.4
0.3
0.2
0.1
0.0



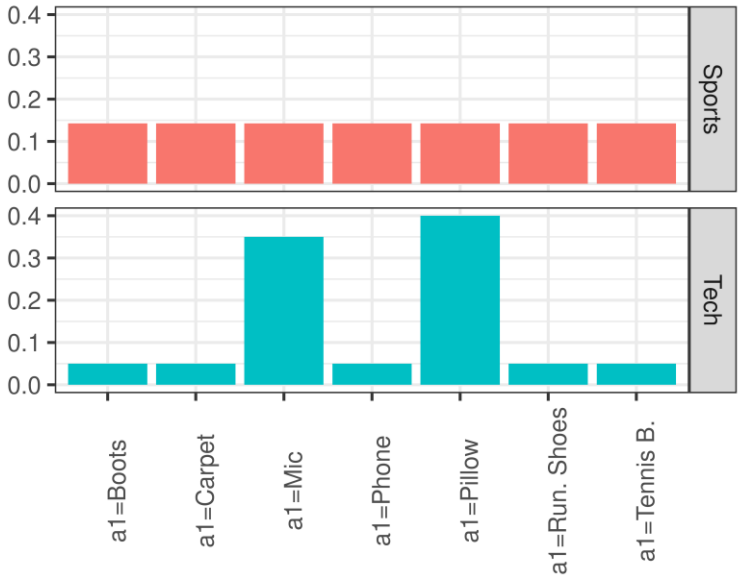
Horvitz-Thompson Based Methods

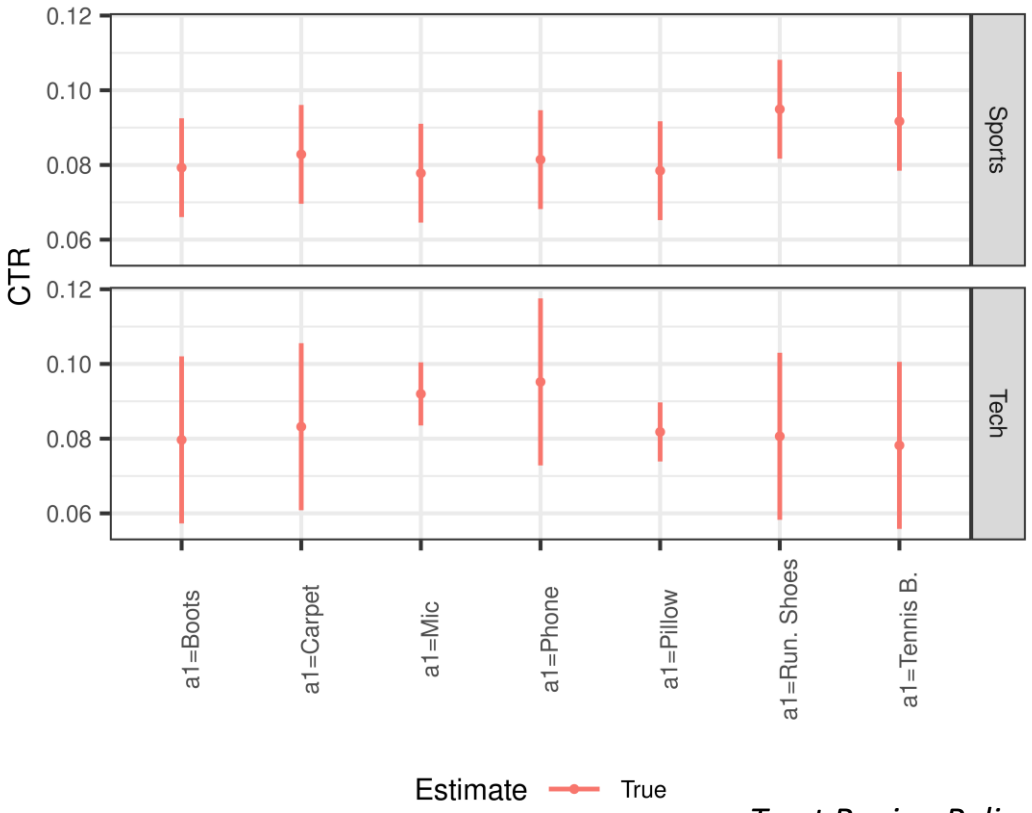
Make bias-variance trade-offs by modifying: $\frac{c_i}{\pi_0(a|\boldsymbol{x}_i)}$

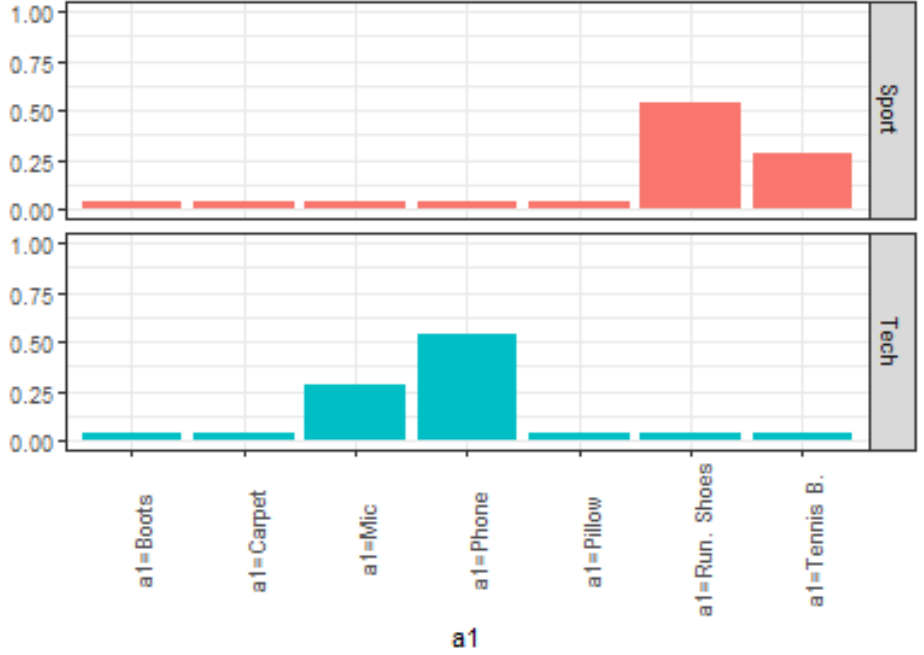
e.g. clipping or Stochastic Variance Penalization. "Policies Cling"

Dudík, Miroslav, et al. 2012.
Bottou, Léon, et al. 2013
Swaminathan, Adith, and Thorsten Joachims. 2015

$\pi_0(a|\boldsymbol{x})$

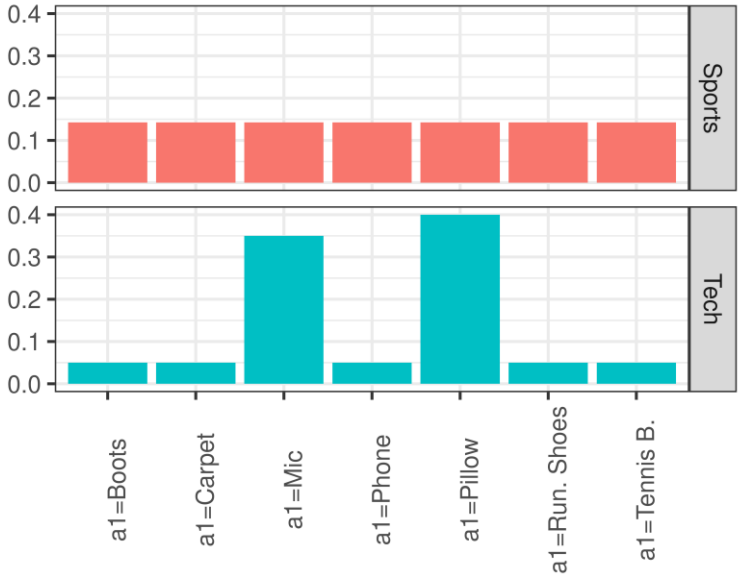






$KL(\pi(\cdot), \pi_0(\cdot)$

$\pi_0(a|\boldsymbol{x})$



Trust Region Policy Optimization

Explicitly holds the new policy to be not too far from the old one..

Schulman, John, et al. "Trust region policy optimization."

Optimizing the MIPS with Policy Learning

$$E[U|\pi(\cdot)] = \sum_{a} R(a, \mathbf{z})\pi(a|\mathbf{z})$$

Works ok if the number of actions is very small

$$\nabla_{\Xi,\beta} E[U|\pi(\cdot)] = \nabla_{\Xi,\beta} \sum_{a} R(a, \mathbf{z})\pi(a|\mathbf{z})$$

Swaminathan, Adith, and Thorsten Joachims. 2015.

Joachims, Thorsten, Adith Swaminathan, and Maarten De Rijke. 2018.

Reinforce Algorithm for MIPS

$$E[R|\pi(\cdot)] = \sum_a R(a, \mathrm{z})\pi(a|\mathrm{z}) = \sum_a R(a, \mathrm{z}) \frac{\exp(f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_a)}{\sum_{\boldsymbol{i}} \exp(f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_i)}$$

$$a' \sim \pi(a|\mathbf{z}), \quad \pi(a|\mathbf{z}) = \frac{\exp\left(f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_a\right)}{\sum_{\boldsymbol{i}} \exp(f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_i)}$$

$$\nabla_{\Xi,\beta} E[R|\pi(\cdot)] = R(a', \mathbf{z})\, \nabla_{\Xi,\beta} \log \pi(a'|\boldsymbol{z})$$

On its own we don't get a speed up for using this algorithm.

From Single Action to Slates

Product
View

Product View

Product View

Product View

Product View

Recommend Slate

Product View

Product View

Recommend Slate

Slate =

$$\text{argsort}(f(v_1 = \text{  } \; v_2 = \text{  })^T \boldsymbol{\beta})_{1:3}$$

Product View

Product View

Recommend Slate

No click

Slate =

$$\text{argsort}(f(v_1 = \boxed{}\ v_2 = \boxed{})^T \boldsymbol{\beta})_{1:3}$$

Product View

Product View

Recommend Slate

Product View

Slate =

$$\text{argsort}(f(v_1 = \text{  } \quad v_2 = \text{  })^T \boldsymbol{\beta})_{1:3}$$

Product View

Product View

Recommend Slate

Product View

Recommend Slate

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $v_3 =$  $)^T \boldsymbol{\beta})_{1:2}$

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $)^T \boldsymbol{\beta})_{1:3}$

Product View

Product View

Recommend Slate

Product View

Recommend Slate

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $v_3 =$  $)^T \boldsymbol{\beta})_{1:2}$

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $)^T \boldsymbol{\beta})_{1:3}$

Product View

Product View

Recommend Slate

Product View

Recommend Slate

No click

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $v_3 =$  $)^T \boldsymbol{\beta})_{1:2}$

Slate = $\text{argsort}(f(v_1 =$  $v_2 =$  $)^T \boldsymbol{\beta})_{1:3}$

Product View

Product View

Recommend Slate

Product View

Recommend Slate

Sale

Slate = argsort($f(v_1 =$  $v_2 =$  $v_3 =$  $)^T \boldsymbol{\beta})_{1:2}$

Slate = argsort($f(v_1 =$  $v_2 =$  $)^T \boldsymbol{\beta})_{1:3}$

Reward Estimation for Slates Using Horvitz-Thompson

Horvitz-Thompson Slate Reward Estimator

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \cdot c \cdot P(c|a_1, .., a_K, \boldsymbol{x}) P(\boldsymbol{x}) dc \; da_1 .. da_K d\boldsymbol{x}$$

Horvitz-Thompson Slate Reward Estimator

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \cdot c \cdot P(c|a_1, .., a_K, \boldsymbol{x}) P(\boldsymbol{x}) dc \; da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \mathbb{E}[c|a_1, \dots, a_K, \boldsymbol{x}] P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

Horvitz-Thompson Slate Reward Estimator

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \cdot c \cdot P(c|a_1, .., a_K, \boldsymbol{x}) P(\boldsymbol{x}) dc \, da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \mathbb{E}[c|a_1, \dots, a_K, \boldsymbol{x}] P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \left(\sum_k \mathbb{E}[r_k|a_k \boldsymbol{x}] \right) P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

Horvitz-Thompson Slate Reward Estimator

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \cdot c \cdot P(c|a_1, .., a_K, \boldsymbol{x}) P(\boldsymbol{x}) dc \, da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \mathbb{E}[c|a_1, \dots, a_K, \boldsymbol{x}] P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \left(\sum_k \mathbb{E}[r_k|a_k \boldsymbol{x}] \right) P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

The fundamental assumption is this.

Li, S., Abbasi-Yadkori, Y., Kveton, B., Muthukrishnan, S., Vinay, V., & Wen, Z. 2018

Horvitz-Thompson Slate Reward Estimator

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \cdot c \cdot P(c|a_1, .., a_K, \boldsymbol{x}) P(\boldsymbol{x}) dc \; da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \mathbb{E}[c|a_1, \dots, a_K, \boldsymbol{x}] P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \left(\sum_k \mathbb{E}[r_k|a_k \boldsymbol{x}] \right) P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int (\sum_k \pi_0(a_k|\boldsymbol{x}) \frac{\pi(a_k|\boldsymbol{x})}{\pi_0(a_k|\boldsymbol{x})} \mathbb{E}[r_k|a_k \boldsymbol{x}]) P(\boldsymbol{x}) \; da_1 .. da_K d\boldsymbol{x}$$

Horvitz-Thompson Slate Reward Estimator

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \cdot c \cdot P(c|a_1, .., a_K, \boldsymbol{x}) P(\boldsymbol{x}) dc \; da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \mathbb{E}[c|a_1, \dots, a_K, \boldsymbol{x}] P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \left(\sum_k \mathbb{E}[r_k|a_k \boldsymbol{x}] \right) P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int (\sum_k \pi_0(a_k|\boldsymbol{x}) \frac{\pi(a_k|\boldsymbol{x})}{\pi_0(a_k|\boldsymbol{x})} \mathbb{E}[r_k|a_k \boldsymbol{x}]) P(\boldsymbol{x}) \; da_1 .. da_K d\boldsymbol{x}$$

Horvitz-Thompson Slate Reward Estimator

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \cdot c \cdot P(c|a_1, .., a_K, \boldsymbol{x}) P(\boldsymbol{x}) dc \; da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \mathbb{E}[c|a_1, \dots, a_K, \boldsymbol{x}] P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \left(\sum_k \mathbb{E}[r_k|a_k \boldsymbol{x}] \right) P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int (\sum_k \pi_0(a_k|\boldsymbol{x}) \frac{\pi(a_k|\boldsymbol{x})}{\pi_0(a_k|\boldsymbol{x})} \mathbb{E}[r_k|a_k \boldsymbol{x}]) P(\boldsymbol{x}) \; da_1 .. da_K d\boldsymbol{x}$$

These are marginal distributions, i.e.

$$\pi_0(a_k|\boldsymbol{x}) = \sum_{a'_1} .. \sum_{a'_{k-1}} \pi_0(a'_1, \dots, a'_{k-1}, a_k|\boldsymbol{x})$$

Horvitz-Thompson Slate Reward Estimator

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \cdot c \cdot P(c|a_1, .., a_K, \boldsymbol{x}) P(\boldsymbol{x}) dc \; da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \mathbb{E}[c|a_1, \dots, a_K, \boldsymbol{x}] P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int \pi_0(a_1, \dots, a_K|\boldsymbol{x}) \frac{\pi(a_1, \dots, a_K|\boldsymbol{x})}{\pi_0(a_1, \dots, a_K|\boldsymbol{x})} \left(\sum_k \mathbb{E}[r_k|a_k \boldsymbol{x}] \right) P(\boldsymbol{x}) da_1 .. da_K d\boldsymbol{x}$$

$$E[U|\pi(\cdot)] = \int (\sum_k \pi_0(a_k|\boldsymbol{x}) \frac{\pi(a_k|\boldsymbol{x})}{\pi_0(a_k|\boldsymbol{x})} \mathbb{E}[r_k|a_k \boldsymbol{x}]) P(\boldsymbol{x}) \; da_1 .. da_K d\boldsymbol{x}$$

These are marginal distributions, i.e.

$$\pi(a_k|\boldsymbol{x}) = \sum_{a'_1} .. \sum_{a'_{k-1}} \pi(a'_1, \dots, a'_{k-1}, a_k|\boldsymbol{x})$$

Product Assumption

Independence assumptions make marginalization cheap/free… but it comes at a cost …

$$\pi_0(a_1, \ldots, a_K|\boldsymbol{x}) = \prod_k \pi_0(a_k|\boldsymbol{x})$$

$$\pi(a_1, \ldots, a_K|\boldsymbol{x}) = \prod_k \pi(a_k|\boldsymbol{x})$$

Product Assumption

Independence assumptions make marginalization cheap/free… but it comes at a cost …

$$\pi_0(a_1, \ldots, a_K|\boldsymbol{x}) = \prod_k \pi_0(a_k|\boldsymbol{x})$$

$$\pi(a_1, \ldots, a_K|\boldsymbol{x}) = \prod_k \pi(a_k|\boldsymbol{x})$$

Reward Estimation for Slates Using A Model

Probabilistic Rank and Reward

Here we outline the Probabilistic Rank and Reward Model (PRR), which has the following properties:

- We assume at most one element in the slate is clicked
  - This differs with IIPS, Position Based Model, Cascade Model etc
- PRR includes user engagement features and recommendation features
- PRR incorporates:
  - was the slate clicked? *bandit approach*
  - which item was clicked? *IIPS, position-based model*
- Gives a direct MIPS

Details in: Aouali et al. 2022

Preference Learning vs Reward

- **Ranking model** (like BPR)
- Trained on clicked-banners only
- Not reward optimizing (it doesn't see it)

Reward model
- Trained on all banners
- Returns calibrated CTR

Preference Learning vs Reward

- **Ranking model** (like BPR)
- Trained on clicked-banners only
- Not reward optimizing (it doesn't see it)

Reward model
- Trained on all banners
- Returns calibrated CTR

PRR combines both!

The Probabilistic Rank and Reward model (our approach)

$$\bar{c}, r_1, \dots, r_K | a_1, \dots, a_K, \mathbf{y}, \mathbf{z} \sim \text{categorical}\left(\frac{\theta_0}{Z}, \frac{\theta_1}{Z}, \dots, \frac{\theta_K}{Z}\right), \qquad Z = \sum \theta_k$$

$$\theta_0 = \exp(\boldsymbol{y}^T \boldsymbol{\phi})$$

$\boldsymbol{y}$ – User engagement features (or bidding features)
$\boldsymbol{\phi}$ – nuisance parameters

The Probabilistic Rank and Reward model (our approach)

$$\bar{c}, r_1, \dots, r_K | a_1, \dots, a_K, \mathbf{y}, \mathbf{z} \sim \text{categorical}\left(\frac{\theta_0}{Z}, \underbrace{\frac{\theta_1}{Z}, \dots, \frac{\theta_K}{Z}}\right), \qquad Z = \sum \theta_k$$

$$\theta_K = \exp(g_\Gamma(\mathbf{z})^T \mathbf{\Psi}_{a_k}) \exp(\gamma_k) + \exp(\alpha_k)$$

$\mathbf{z}$ – context
$g_\Gamma(\mathbf{z})$ – contextual embedding
Ψ_a – Item embedding for item *a*
γ, α – multiplicative and additional position bias

Reminder

$$a_1, \dots, a_K = \text{argsort}(f_\Xi(\mathbf{z})^T \boldsymbol{\beta})$$

$$f_\Xi = g_\Gamma \text{ and } \boldsymbol{\beta} = \mathbf{\Psi}$$

Reward and Rank only variants

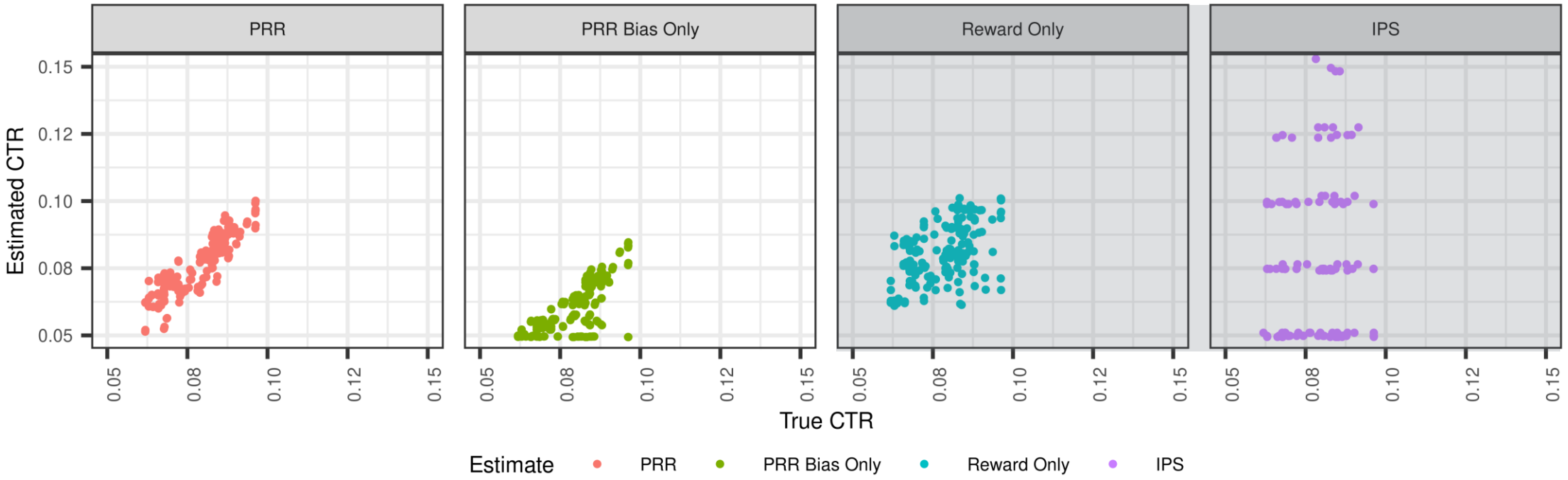
Reward model $$\bar{c}, c|\ a_1, \dots, a_K, y, z \sim \text{categorical}\left(\frac{\theta_0}{Z}, \sum \frac{\theta_k}{Z}\right), \qquad Z = \sum_{k=0}^{K} \theta_k$$

Ranking model $$r_1, \dots, r_K | c = 1, a_1, \dots, a_K, y, z \sim \text{categorical}\left(\frac{\theta_1}{Z}, \dots, \frac{\theta_K}{Z}\right), \qquad Z = \sum_{k=1}^{K} \theta_k$$

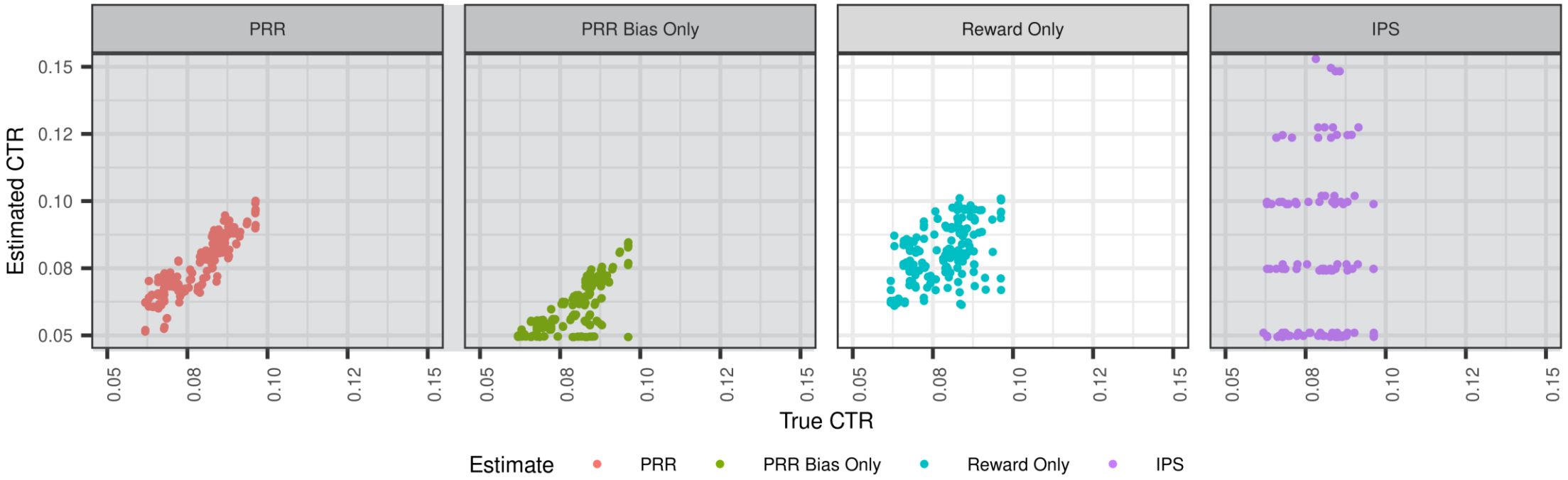
$$\theta_0 = \exp(\boldsymbol{y}^T \boldsymbol{\phi})$$

$$\theta_K = \exp(g_\Gamma(\mathbf{z})^T \boldsymbol{\Psi}_{ak}) \exp(\gamma_k) + \exp(\alpha_k)$$

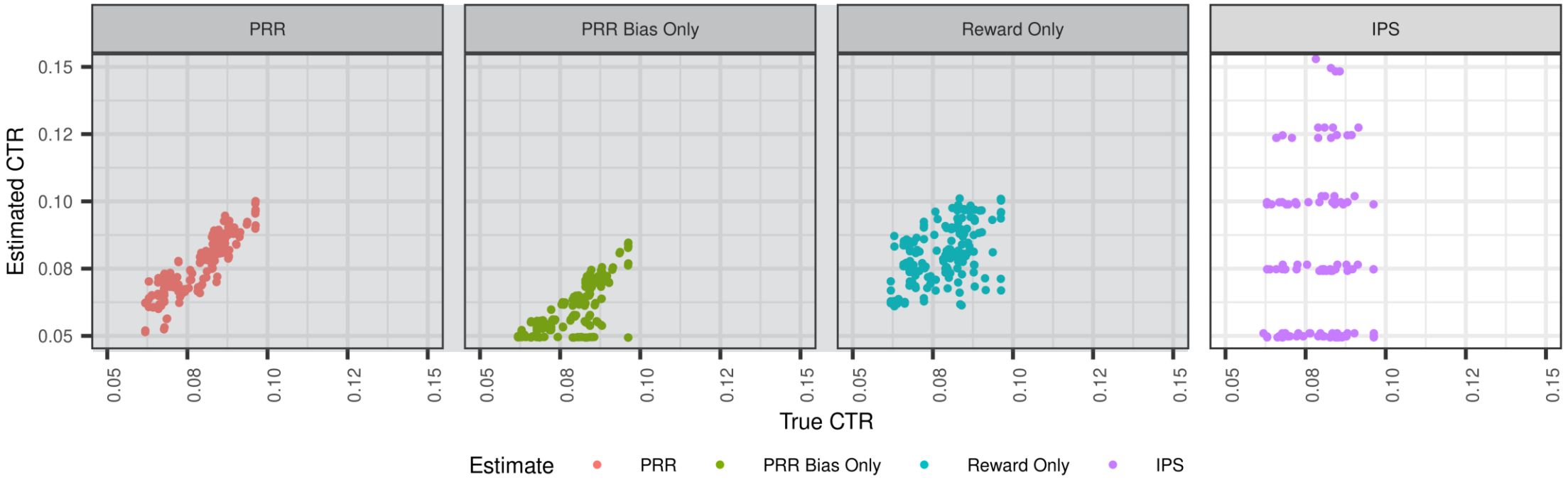
Calibration on toy example



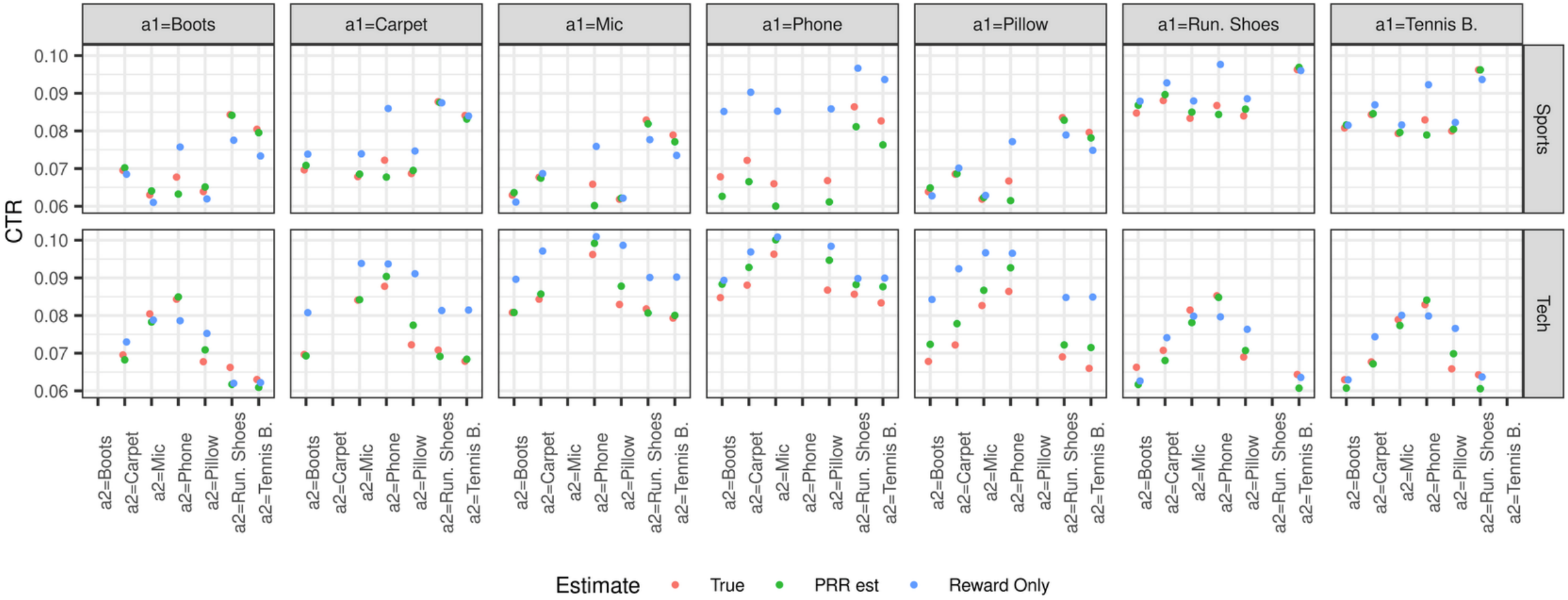
Calibration on toy example



Calibration on toy example



Click-Through Rate Estimates



Slate Recommendations

Decision Rule Optimization

Reinforce Algorithm for Slates

$$\mathbb{E}[U|\pi(\cdot)] = \sum_{a_1} \dots \sum_{a_K} R(a_1, .., a_K, \mathbf{z})\pi(a_1, .., a_K|\mathbf{z})$$

$$\nabla_{\Xi,\beta}\mathbb{E}[U|\pi(\cdot)] = \nabla_{\Xi,\beta} \sum_{a_1} \dots \sum_{a_K} R(a_1, .., a_K, \mathbf{z})\pi(a_1, .., a_K|\mathbf{z})$$

This is now totally infeasible

Reinforce Algorithm for Slates

$$\mathbb{E}[U|\pi(\cdot)] = \sum_{a} R(a_1, \dots a_K, \mathbf{x})\pi(a_1, \dots a_K|\mathbf{x})$$

$$a'_1, \dots, a'_K \sim \pi(\cdot\,|\mathbf{x})$$

$$\nabla_{\Xi,\beta}\mathbb{E}[U|\pi(\cdot)] = R(a'_1, \dots, a'_K, \mathbf{x})\nabla_{\Xi,\beta}\log\pi(a'_1, \dots, a'_K|\mathbf{x})$$

If we can sample and evaluate $\pi(\cdot\,|\mathbf{x})$ then we can proceed…

Independence Assumption

$$\pi(a_1, \dots, a_K|\boldsymbol{x}) = \prod_k \pi(a_k|\boldsymbol{x})$$

What if the best two recommendations are rice and beer

Firstly, we don't want:

$$\pi(a = \text{rice}|\boldsymbol{x}) = 1$$

as this forces duplication for large slates i.e.

$$\pi(a_1 = \text{rice}, a_2 = \text{rice}|\boldsymbol{x}) = 1$$

$$f_{\Xi}(\mathbf{z})^T \boldsymbol{\beta}_{\text{rice}} \gg f_{\Xi}(\mathbf{z})^T \boldsymbol{\beta}_{\mathbf{i}}, \forall_{i \neq \text{rice}}$$

Independence Assumption

$$\pi(a_1, \ldots, a_K|\boldsymbol{x}) = \prod_k \pi(a_k|\boldsymbol{x})$$

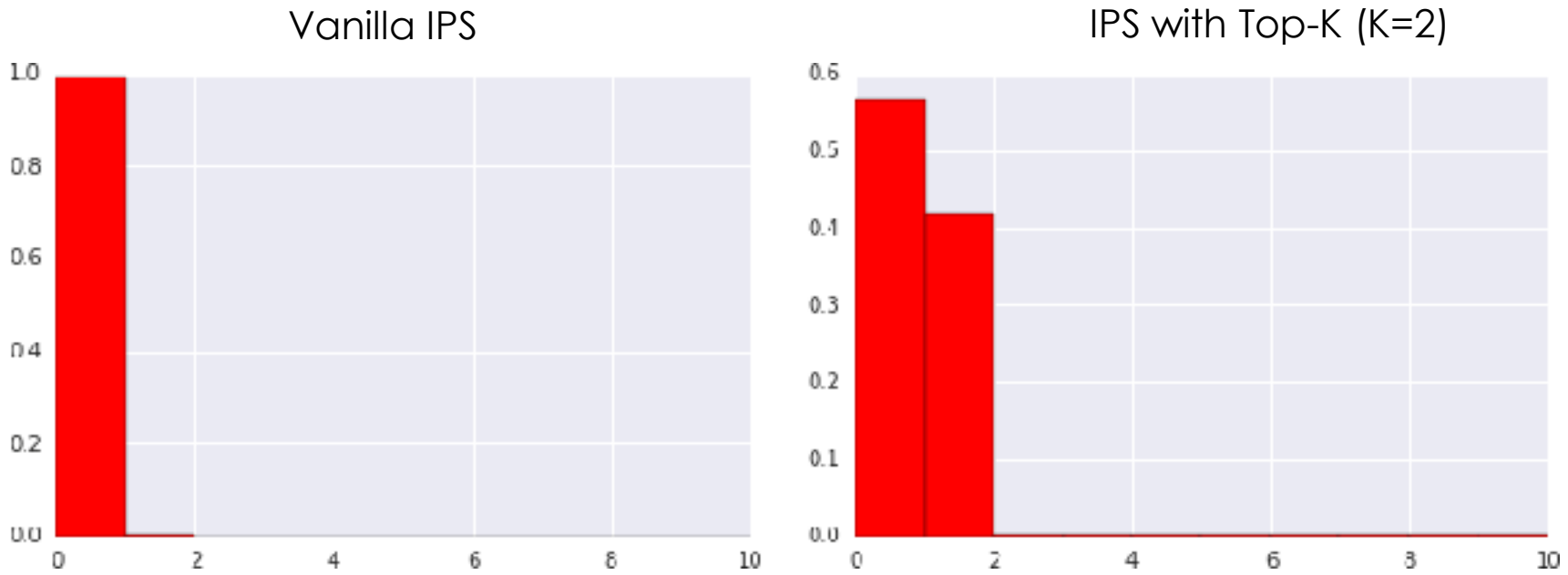
What if the best two recommendations are rice and beer

Firstly, we don't want:

$$\pi(a = \text{rice}|\boldsymbol{x}) = 1$$

Chen, 2019 proposes a solution to this which keeps the probability is spread out over the Top-K items.

Independence Assumption With Top-K



From Top-K Off-Policy Correction for a REINFORCE Recommender System [1]

The Top-K correction modifies the loss to stop $\pi(a_k|x)$ converging to a degenerate distribution

$$\pi(a_1, \ldots, a_K|\mathbf{x}) = \Pi\, \pi(a_k|\mathbf{x})$$

Chen, et al. 2019

Independence Assumption

$$\pi(a_1, \ldots, a_K|\boldsymbol{x}) = \prod_k \pi(a_k|\boldsymbol{x})$$

What if the best two recommendations are rice and beer

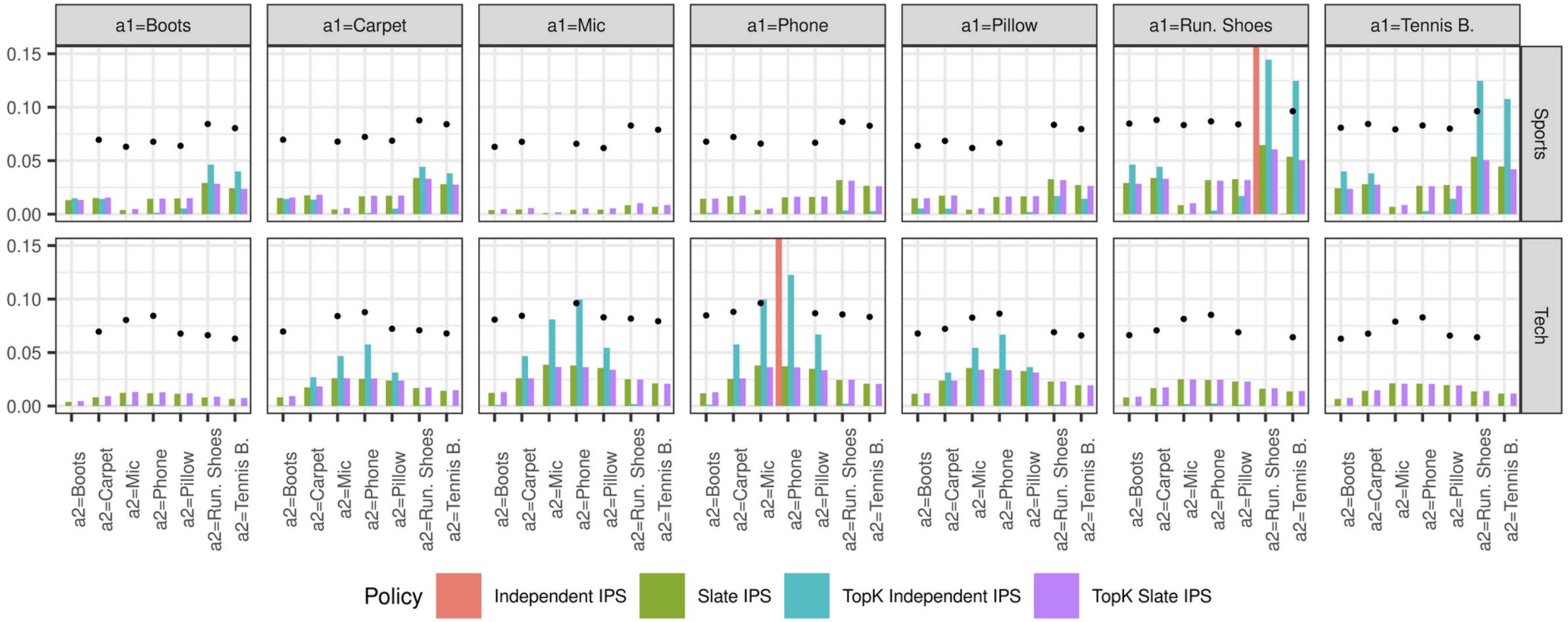
$$\pi(a = \text{rice}|\boldsymbol{x}) = 0.51$$

$$\pi(a = \text{beer}|\boldsymbol{x}) = 0.49$$

Then

$$\pi(a_1 = \text{rice}, a_2 = \text{rice}|\boldsymbol{x}) = 0.2601$$
$$\pi(a_1 = \text{rice}, a_2 = \text{beer}|\boldsymbol{x}) = 0.2499$$
$$\pi(a_1 = \text{beer}, a_2 = \text{rice}|\boldsymbol{x}) = 0.2499$$
$$\pi(a_1 = \text{beer}, a_2 = \text{beer}|\boldsymbol{x}) = 0.2401$$

Policy Learning



Placket-Luce Policy Formulation

Sampling without replacement is called Placket-Luce

$$\pi(a_1, \dots, a_K|\mathbf{z}) = \frac{e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_{a_1}}}{\sum e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_i}} \cdot \frac{e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_{a_2}}}{e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_{a_1}} - \sum e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_i}} \dots \frac{e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_{a_K}}}{e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_{a_1}} + .. + e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_{a_{K-1}}} - \sum e^{f_\Xi(\mathbf{z})^T \boldsymbol{\beta}_i}}$$

Reinforce Algorithm for MIPS

$$E[U|\pi(\cdot)] = \sum_{a} R(a_1, \dots a_K, \mathbf{x})\pi(a_1, \dots a_K|\mathbf{x})$$

$$a'_1, \dots, a'_K \sim \pi(\cdot\,|\mathbf{x})$$

$$\nabla_{\Xi,\beta} E[U|\pi(\cdot)] = R(a'_1, \dots, a'_K, \mathbf{x}) \log \nabla_{\Xi,\beta} \pi(a'_1, \dots, a'_K|\mathbf{x})$$

Simulated A/B Tests – Slate Sizes

Algorithm	K=2	K=6	K=18
Oracle	72.62 ± 0.45%	64.43 ± 0.48%	27.08 ± 0.44%
Random	3.39 ± 0.18%	7.72 ± 0.27%	5.24 ± 0.22%
PRR	**50.47 ± 0.5%**	**61.70 ± 0.49%**	**26.7 ± 0.44%**
PRR (Bias Only)	36.0 ± 0.48%	60.12 ± 0.49%	25.44 ± 0.44%
Ranking Model	35.79 ± 0.48 %	48.88 ± 0.5%	**26.31 ± 0.44%**
Reward Model	42.04 ± 0.49%	22.28 ± 0.42%	5.43 ± 0.23%
Slate IPS	20.02 ± 0.4%	26.67 ± 0.44%	3.31 ± 0.18%
Top-K Slate IPS	20.02 ± 0.4%	17.84 ± 0.38%	3.31 ± 0.18%
IIPS	40.42 ± 0.49%	48.91 ± 0.5%	25.36 ± 0.44%
Top-K IIPS	40.86 ± 0.49%	52.77 ± 0.5%	25.6 ± 0.44%

Table 3: Simulated A/B test with varying slate size. $P = 1k$, $N_{\text{Train}} = 10k$

Simulated A/B Tests – Scaling

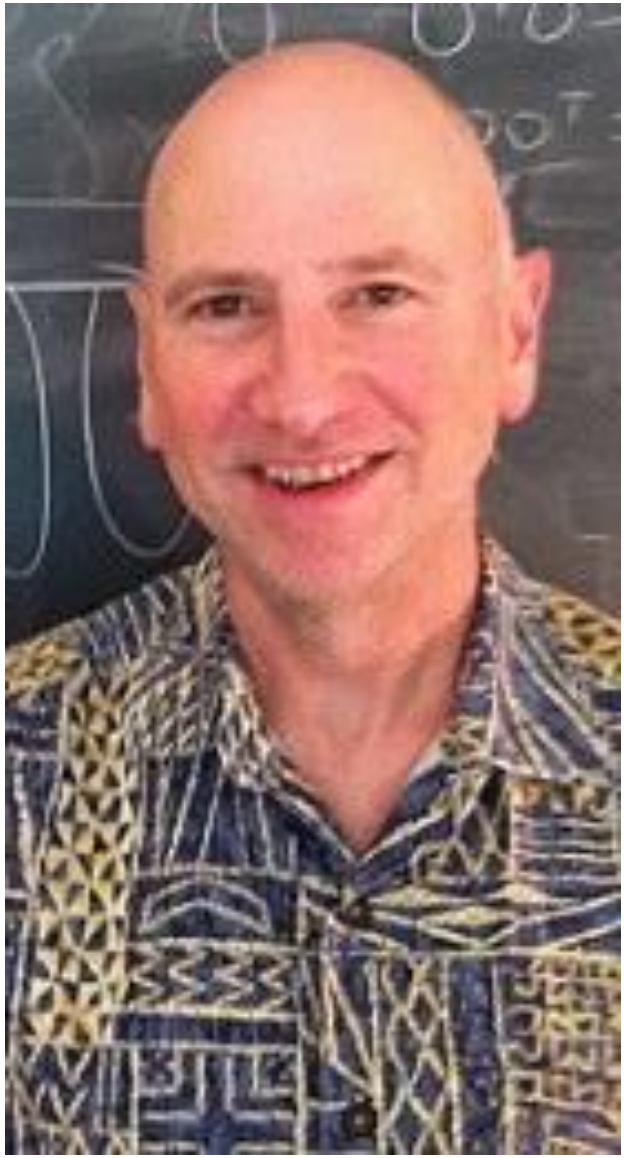
Algorithm	P=1k	P=5k	P=10k
Oracle	49.21 ± 0.50%	76.94 ± 0.42%	78.01 ± 0.41%
Random	3.65 ± 0.19%	7.0 ± 0.26%	5.68 ± 0.23%
PRR	**42.63 ± 0.49%**	**58.01 ± 0.49%**	**57.18 ± 0.49%**
PRR (Bias only)	**42.88 ± 0.49%**	49.98 ± 0.5%	44.31 ± 0.5%
Ranking Model	26.15 ± 0.44%	50.57 ± 0.5%	49.72 ± 0.5%
Reward Model	2.83 ± 0.17%	38.35 ± 0.49%	19.14 ± 0.39%
Slate IPS	20.93 ± 0.41%	29.19 ± 0.45%	18.01 ± 0.38%
Top-K Slate IPS	27.23 ± 0.45%	25.38 ± 0.44%	15.76 ± 0.36%
IIPS	36.56 ± 0.48%	51.47 ± 0.5 %	34.85 ± 0.48%
Top-K IIPS	**43.01 ± 0.5%**	53.96 ± 0.5%	37.24 ± 0.48%

Table 4: Simulated A/B test with varying catalog size. $K = 5$, $N_{\text{Train}} = 10k$

Simulated A/B Tests – 1M Products

Algorithm	CTR	Time/epoch (s)
Oracle	94.84 ± 0.31%	-
Random	6.25 ± 0.34%	-
PRR	**67.19 ± 0.66%**	4.36
PRR (Bias only)	46.87 ± 0.71%	4.36
Ranking Model	52.13 ± 0.71%	**2.81**
Reward Model	31.37 ± 0.66%	4.37
Slate IPS	3.11 ± 0.25%	68.38
Top-K SlateIPS	3.11 ± 0.25%	69.81
IIPS	42.22 ± 0.7%	68.60
Top-K IIPS	48.58 ± 0.71%	69.56

Table 6: Simulated A/B test for 1M products. $N_{\text{Train}} = 10k$, $K = 5$, **batch size 516. Note that Ranking and IPS methods are only using 6% of the dataset.**



A Great Controversy in Statistics

Christopher Sims and Larry Wasserman had an “epic debate” on the Horvitz-Thompson estimator

Around if it causes a problem for likelihood/Bayesian inference

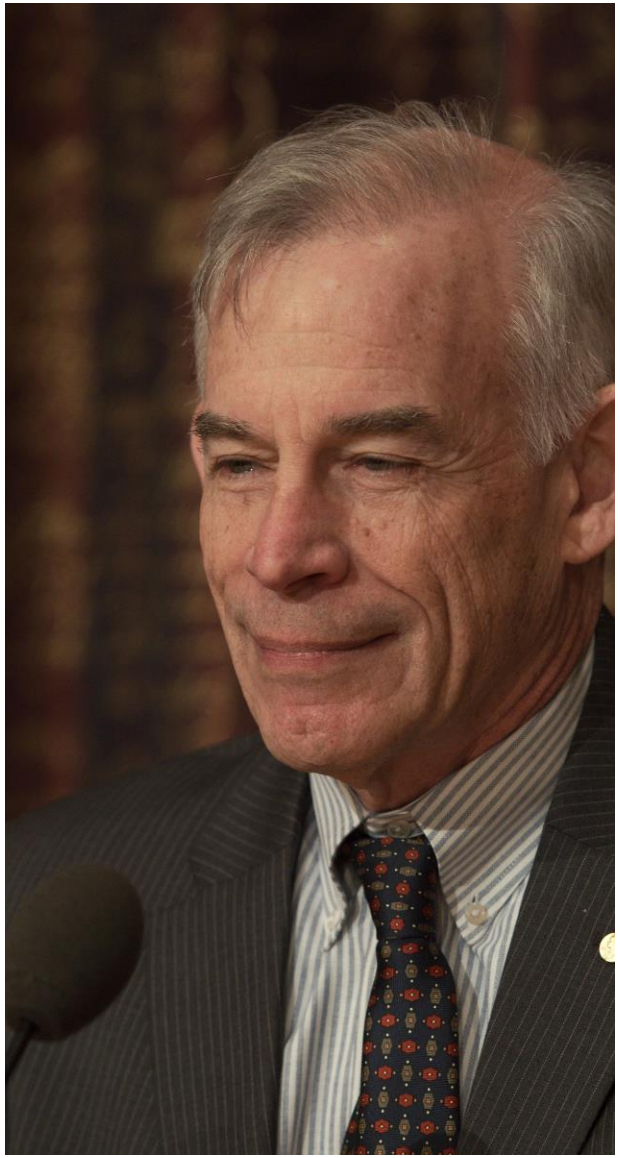
Bayesians/Likelihoodists should not use the propensity score..

PRR + BLOB ignore the propensity score and outperform other methods

Debate and discussion:

Bayesian Causal Inference for Real World Interactive Systems (bcirwis2021.github.io)

Christopher Sims - Large Parameter Spaces and Weighted Data: A Bayesian Perspective - YouTube



Notebook 1 (return to)

https://colab.research.google.com/github/otmhi/Reward-Optimizing-Reco/blob/main/Reward_Optimizing_Slate_Recommendation_with_DL_and_MIPS_Part_1.ipynb

“Technology”: SNIPS Reinforce

Optimizing the Decision Rule Quickly

Optimizing the objective

Our objective is:

$$U_{\Xi,\beta}(\boldsymbol{x}) = \mathbb{E}_{a\sim\pi_{\Xi,\beta}(.|\boldsymbol{x})}[R(a,\boldsymbol{x})]$$

$$\nabla_{\Xi,\beta} U_{\Xi,\beta}(\boldsymbol{x}) = \mathbb{E}_{a\sim\pi_{\Xi,\beta}(.|\boldsymbol{x})}[R(a,\boldsymbol{x})\nabla_{\Xi,\beta}\log\pi_{\Xi,\beta}(a|\boldsymbol{x})]$$

There are mainly two problems with this gradient :

- $\log\pi_\theta(a|x)$ involves the computation of the normalizing constant ➔ O(P)
- If we want to compute the exact expectation ➔ O(P) (Infeasible)
- If we want to approximate the gradient by sampling ➔ O(P)

Everything scales **at least linearly in P** which can be very costly in large catalog sizes.

Policy Optimization is not Maximum Likelihood

$$\log P(a|\boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\Xi}) = f_{\boldsymbol{\Xi}}(\boldsymbol{x})^T \boldsymbol{\beta}_a - \log \sum_{a'} \exp(f_{\boldsymbol{\Xi}}(\boldsymbol{x})^T \boldsymbol{\beta}_{a'})$$

$$\nabla_{\beta,\boldsymbol{\Xi}} \log P(a|\boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\Xi}) = \nabla_{\beta,\boldsymbol{\Xi}} f_{\boldsymbol{\Xi}}(\boldsymbol{x})^T \boldsymbol{\beta}_a - \mathbb{E}_{P(a'|\boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\Xi})}[\nabla_{\beta,\boldsymbol{\Xi}} f_{\boldsymbol{\Xi}}(\boldsymbol{x})^T \boldsymbol{\beta}_{a'}]$$

Lots of work on how to solve this problem! e.g.

Bengio, Yoshua, and Jean-Sébastien Senécal. 2003.

Ruiz, Francisco, et al. 2018.

Rawat, Ankit Singh, et al. 2019.

Optimizing the objective

Our objective is:

$$U_{\Xi,\beta}(\boldsymbol{x}) = \mathbb{E}_{A\sim\pi_{\Xi,\beta}(.|\boldsymbol{x})}[R(a,\boldsymbol{x})]$$

$$\nabla_{\Xi,\beta}U_{\Xi,\beta}(\boldsymbol{x}) = \mathbb{E}_{a\sim\pi_{\Xi,\beta}(.|\boldsymbol{x})}[R(a,\boldsymbol{x})\nabla_{\Xi,\beta}\log\pi_{\Xi,\beta}(a|\boldsymbol{x})]$$

$$\nabla_{\Xi,\beta}U_{\Xi,\beta}(x) = Cov_{a\sim\pi_{\theta}(.|\boldsymbol{x})}[R(a,\boldsymbol{x}),\nabla_{\Xi,\beta}\exp f_{\Xi}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{\beta}_a]$$

With $Cov[Y,\ \boldsymbol{Z}] = \mathbb{E}[(Y - \mathbb{E}[Y])(\boldsymbol{Z} - \mathbb{E}[\boldsymbol{Z}])]$

.

Proposal Selection

Additional Assumption

$$\pi(a|x) = \frac{\exp(f_{\Xi}(\boldsymbol{x})^T \boldsymbol{\beta})}{\sum_{a'} \exp(f_{\Xi}(\boldsymbol{x})^T \boldsymbol{\beta}_{a'})}$$

Assume $\boldsymbol{\beta}$ is fixed and we only optimize $f_{\Xi}(\boldsymbol{x})^T$. This structure allows fast search/argsort *O(logP).*

MIPS give us $\alpha_S(x) = \text{argsort}(f_{\Xi}(\boldsymbol{x})^T \boldsymbol{\beta})_{[1,\dots,S]}$ the top (approximate) items of the policy

We define :

$$Q_{\epsilon,K}(a|\boldsymbol{x}) = \begin{cases} \frac{\epsilon}{P} + (1 - \epsilon)\kappa(a|\boldsymbol{x}) \; if \; a \in \alpha_K(\boldsymbol{x}) \\ \frac{\epsilon}{P} \; else \end{cases} \quad \text{with } \kappa_\theta(a|\boldsymbol{x}) = \frac{\exp(f_{\Xi}(\boldsymbol{x})^{\boldsymbol{T}} \boldsymbol{\beta}_a)}{\sum_{a' \in \alpha_K(x)} \exp(f_{\Xi}(\boldsymbol{x})^{\boldsymbol{T}} \boldsymbol{\beta}_{a'})}$$

Self Normalized Importance Sampling

$$\tilde{\pi}(a|\boldsymbol{x}) = \exp(f_{\Xi}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{\beta}_a)$$

$$Q_{\epsilon,K}(a|X) = \begin{cases} \frac{\epsilon}{P} + (1 - \epsilon)\kappa(a|\boldsymbol{x}) \; if \; a \in \alpha_K(\boldsymbol{x}) \\ \frac{\epsilon}{P} \text{ else} \end{cases} \quad \text{with } \kappa_{\Xi,\beta}(a|x) = \frac{\exp(f_{\Xi}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{\beta}_a)}{\sum_{a\prime\in\alpha_K(x)} \exp(f_{\Xi}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{\beta}_{a\prime})}$$

$$\widetilde{w}_i = \frac{\tilde{\pi}(A_i|\boldsymbol{x})}{Q(A_i|\boldsymbol{x})}, \qquad w_i = \frac{\widetilde{w}_i}{\sum \widetilde{w}_j}$$

$$\bar{Y} = \sum_{i=1}^{N} w_i Y_i \,, \qquad \bar{Z} = \sum_{i=1}^{N} w_i Z_i$$

$$\nabla_{\Xi,\beta} U_{\Xi,\beta}(x) = Cov_{a\sim\pi_{\Xi,\beta}(.|\boldsymbol{x})}[R(a,\boldsymbol{x}), \nabla_{\Xi,\beta} \exp f_{\Xi}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{\beta}_a],$$

with $Cov[Y, \; \boldsymbol{Z}] = \mathbb{E}[(Y - \mathbb{E}[Y])(\boldsymbol{Z} - \mathbb{E}[\boldsymbol{Z}])]$

return $\widehat{SNIS}_N = \sum_{i=1}^{N} w_i (Y_i - \bar{Y})(Z_i - \bar{Z})$

Experiments

We test the method on a session completion task : X observed session, Y complementary items.

This can be cast into an offline bandit framework where our policy π_θ takes the observed part X as a context, recommends an item a and receives the binary reward $\hat{r}(a, X) = 1[a \in Y]$.

We use X to build item embeddings $\boldsymbol{\beta}$ of dimension $L \ll P$.
Define x_{emb} as the item embeddings average in session X.

$$\pi(a|x) = \frac{\exp(f_\Xi(\boldsymbol{x})^T \boldsymbol{\beta}_a)}{\sum_{a'} \exp(f_\Xi(\boldsymbol{x})^T \boldsymbol{\beta}_{a'})}$$

	Catalog size	Number of users
Twitch	750K	500K
GoodReads	1.23M	300K

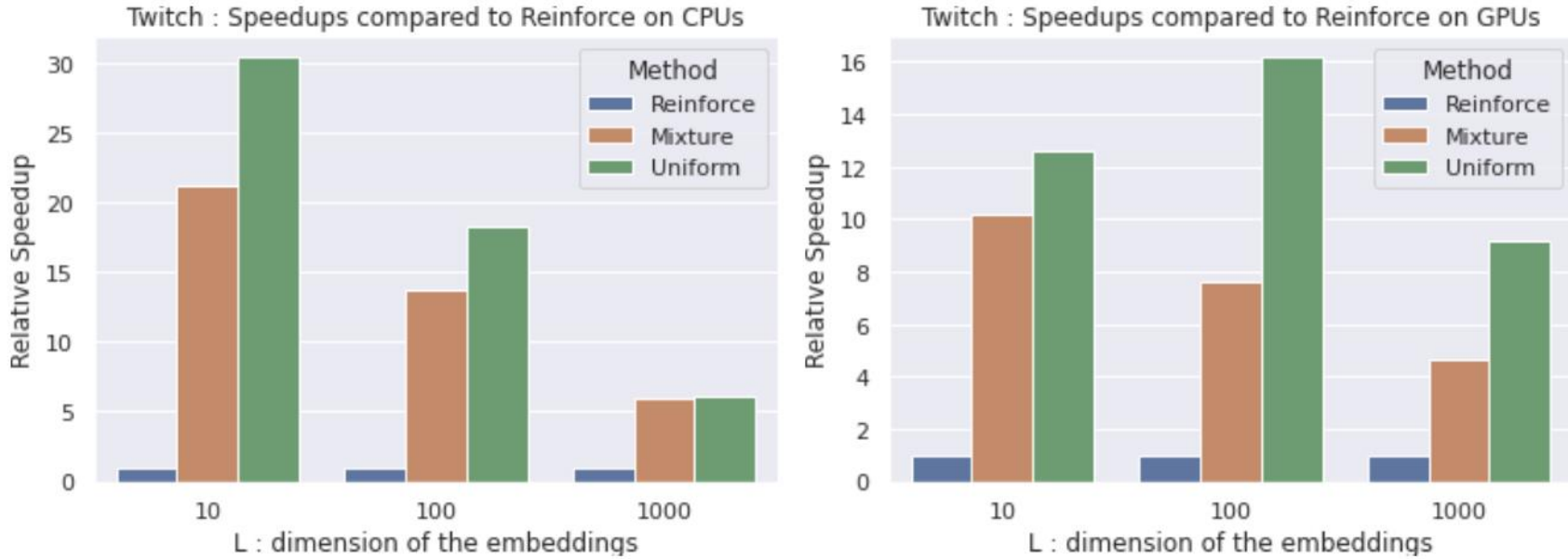
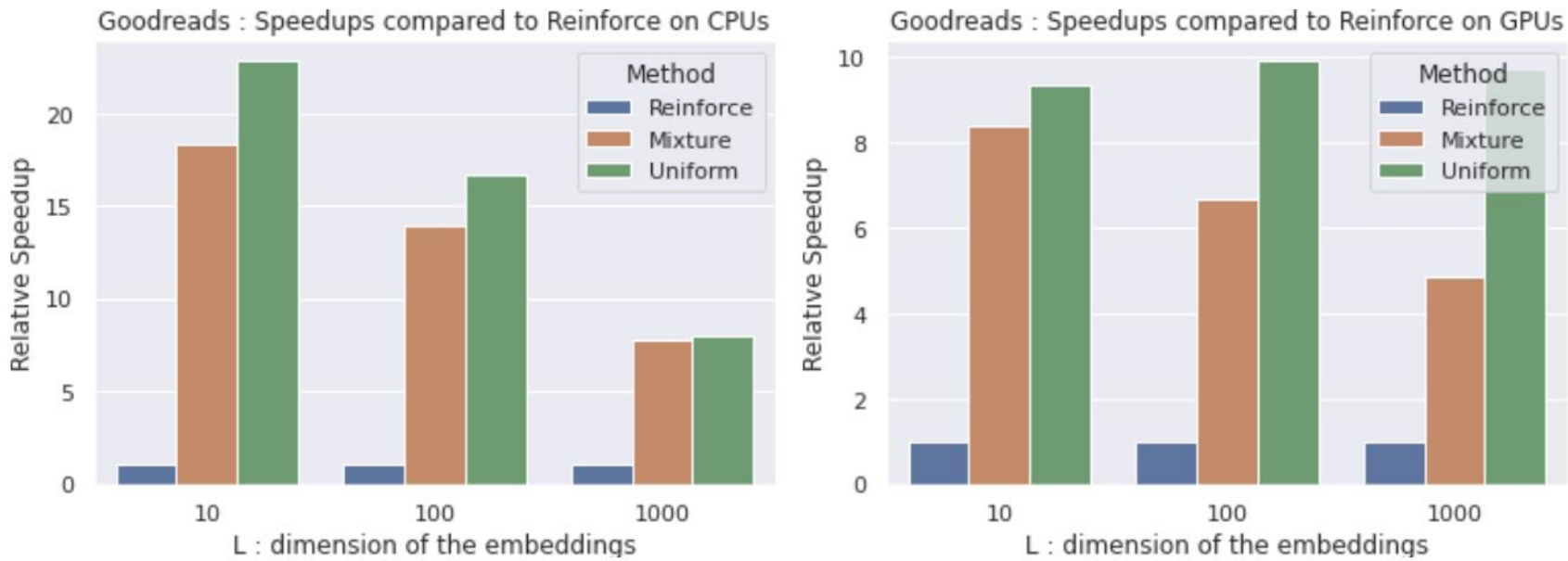


Fig. 1. The Twitch Dataset : The speedup of the proposed algorithms on both CPU and GPU devices



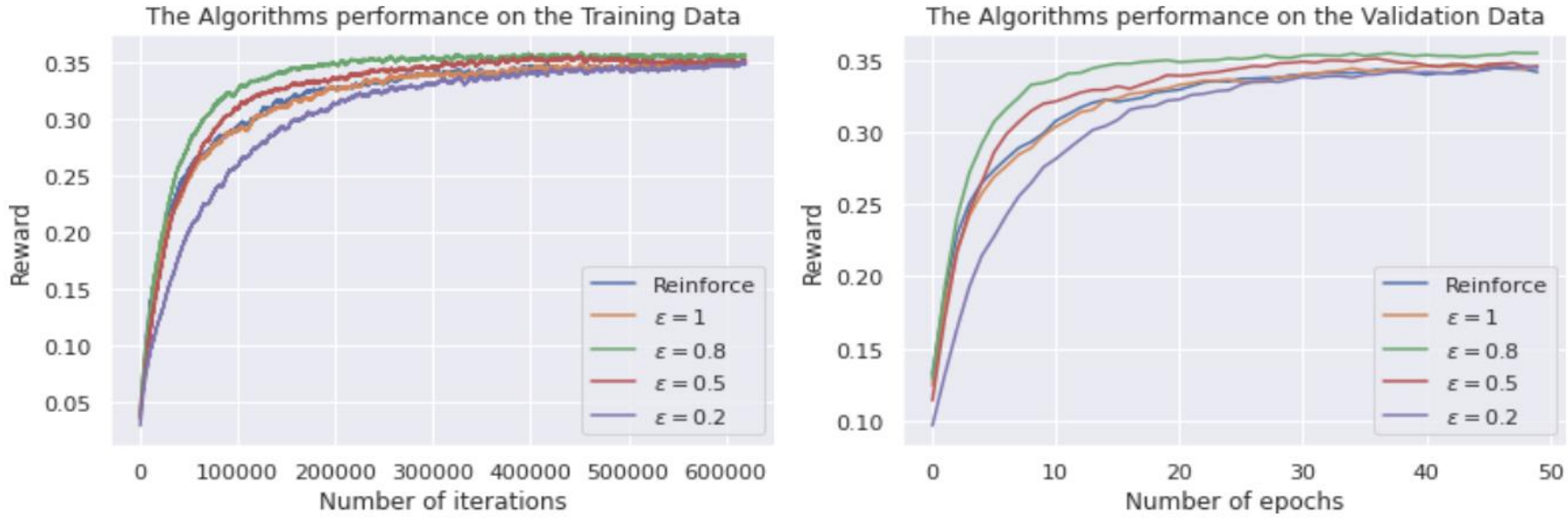


Fig. 3. The Twitch Dataset : ϵ effect on the policy training and validation

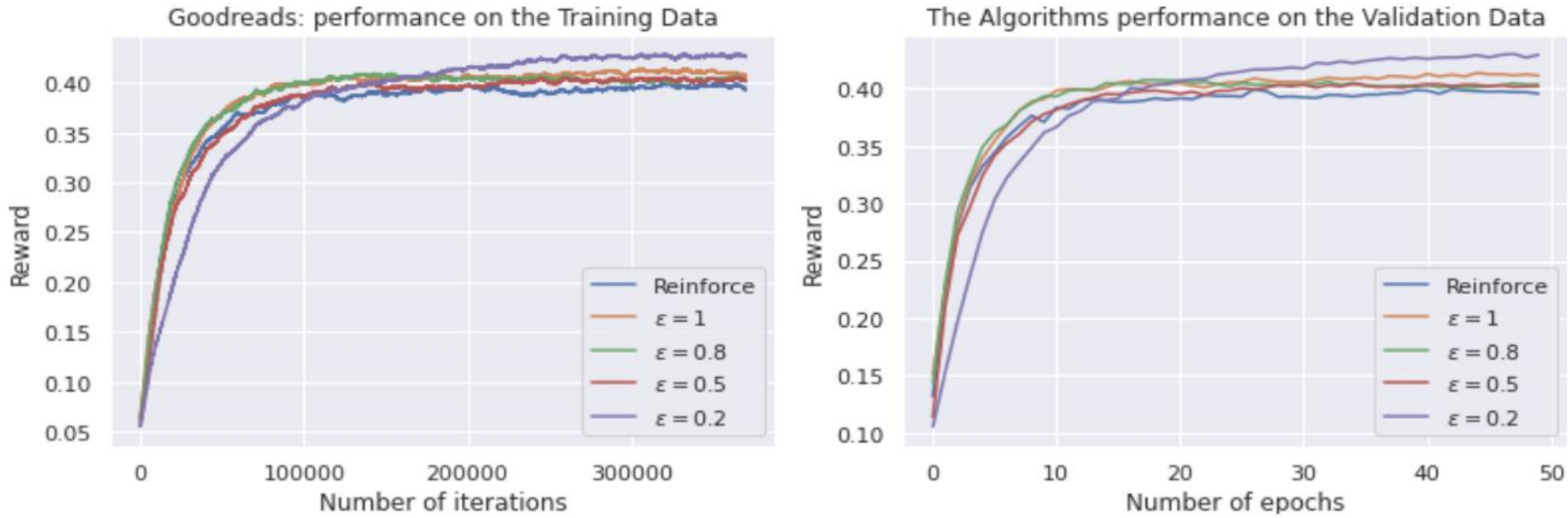


Fig. 4. The Goodreads Dataset : ϵ effect on the policy training and validation

Fig. 5. Performance plots given time : ϵ's effect on the training

Sakhi et al. 2022

Notebook 2

https://colab.research.google.com/github/otmhi/Reward-Optimizing-Reco/blob/main/Reward_Optimizing_Slate_Recommendation_with_DL_and_MIPS_Part_2.ipynb

Tutorial covering Horvitz Thompson for Recommender Systems

RecSys2021 Tutorial